

PARAMETIRC SIMLUATION  
OF  
TACTICAL SINGLE CHANNEL  
FREQUENCY MODULATED COMMUNICATIONS

Philip A. Olson Jr.



# NAVAL POSTGRADUATE SCHOOL

## Monterey, California



# THESIS

Parametric Simulation  
of  
Tactical Single Channel  
Frequency Modulated Communications

by

Philip A. Olson Jr.

September 1980

Thesis Advisor:

A. L. Schoenstadt

Approved for public release; distribution unlimited.

T197053



REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle)  Parametric Simulation of Tactical Single Channel Frequency Modulated Communications		5. TYPE OF REPORT & PERIOD COVERED  Master's Thesis (Sep 80)
6. AUTHOR(s)  Philip Andrew Olson Junior		7. PERFORMING ORG. REPORT NUMBER(s)
8. PERFORMING ORGANIZATION NAME AND ADDRESS  Naval Postgraduate School Monterey, California 93940		9. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
10. CONTROLLING OFFICE NAME AND ADDRESS  Naval Postgraduate School Monterey, California 93940		11. REPORT DATE  September 1980
12. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)  Naval Postgraduate School Monterey, California 93940		13. NUMBER OF PAGES  148
14. SECURITY CLASS. (of this report)		15. DECLASSIFICATION/DOWNGRADING SCHEDULE  Unclassified
16. DISTRIBUTION STATEMENT (of this Report)  Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)  STAR Tactical Communications Communications Simulation Movement Decision Logic Communications		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)  This thesis presents a stochastic simulation model of single channel FM communications that is designed to be used in conjunction with the Simulation of Tactical Alternative Responses (STAR) Combat Model. The communications modelled, the assumptions made, and the interface requirements necessary for inclusion of the Communications Model in the STAR Combat Model are explained in detail. The computer		



code that is used to execute the Communications Model is included in the appendices. An overview is given of those portions of the basic STAR Model with which the Communications Model interfaces directly to provide the reader with sufficient background for the discussion of the Communications Model. The communications input requirements of the model, a definition of the purpose of each routine and event, and definitions of each global variable, set, and entity are provided so that this thesis can serve as a user's manual for the Communications Model.



Approved for Public Release: Distribution Unlimited

Parametric Simulation of Tactical Single Channel  
Frequency Modulated Communications

by

Philip Andrew Olson, Junior  
Major, United States Army  
BS, Principia College, 1968

Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL

September 1980



## ABSTRACT

This thesis presents a stochastic simulation model of single channel FM communications that is designed to be used in conjunction with the Simulation of Tactical Alternative Responses (STAR) Combat Model. The communications modelled, the assumptions made, and the interface requirements necessary for inclusion of the Communications Model in the STAR Combat Model are explained in detail. The computer code that is used to execute the Communications Model is included in the appendices. An overview is given of those portions of the basic STAR Model with which the Communications Model interfaces directly to provide the reader with sufficient background for the discussion of the Communications Model. The communications input requirements of the model, a definition of the purpose of each routine and event, and definitions of each global variable, set, and entity are provided so that this thesis can serve as a user's manual for the Communications Model.



## TABLE OF CONTENTS

DOCUMENTATION PAGE -----	1
APPROVAL -----	3
ABSTRACT -----	4
TABLE OF CONTENTS -----	5
LIST OF FIGURES -----	7
ACKNOWLEDGEMENT -----	8
I. INTRODUCTION -----	9
II. DESCRIPTION OF THE STAR MODEL -----	13
III. COMMUNICATIONS ASSUMPTIONS OF STAR -----	18
IV. DESCRIPTION OF THE PROBLEM -----	20
V. ASSUMPTIONS OF THE COMMUNICATIONS MODULE -----	22
VI. DESCRIPTION OF THE COMMUNICATIONS MODULE -----	25
VII. COMMUNICATIONS MODULE PARAMETERIZATION -----	53
VIII. COMMUNICATIONS MODULE INPUT REQUIREMENTS -----	55
IX. FUTURE MODEL ENHANCEMENTS -----	59
APPENDIX A STAR ROUTINES, EVENTS, AND ARRAYS -----	61
APPENDIX B COMMUNICATIONS ROUTINES AND EVENTS -----	66
APPENDIX C COMMUNICATIONS GLOBAL VARIABLES -----	70
APPENDIX D COMMUNICATIONS PERMANENT ENTITIES -----	73
APPENDIX E COMMUNICATIONS TEMPORARY ENTITIES -----	77
APPENDIX F COMMUNICATIONS ARRAYS -----	80
APPENDIX G COMMUNICATIONS SETS -----	86
APPENDIX H ROUTINE DECISION -----	87
APPENDIX I ROUTINE DEC.CO -----	91
APPENDIX J ROUTINE DEC.BN -----	93
APPENDIX K ROUTINE DEC.BDE -----	97
APPENDIX L ROUTINE DEC.DIV -----	101
APPENDIX M ROUTINE EXIT -----	102



APPENDIX N	ROUTINE GEN.MOVE.DECISION.MSG	-----	103
APPENDIX O	ROUTINE COM.MAIN	-----	104
APPENDIX P	ROUTINE BUILDNETS	-----	105
APPENDIX Q	ROUTINE CHECKNETS	-----	107
APPENDIX R	ROUTINE BNNTBL	-----	108
APPENDIX S	ROUTINE COMMGEAR	-----	109
APPENDIX T	ROUTINE COMMCHECK	-----	111
APPENDIX U	ROUTINE INITIALIZE	-----	113
APPENDIX V	ROUTINE STAT.DUMP	-----	114
APPENDIX W	ROUTINE SSTAT.DUMP	-----	115
APPENDIX X	EVENT COMMO.ATTEMPT	-----	117
APPENDIX Y	ROUTINE TECH.COMMO	-----	121
APPENDIX Z	EVENT CHANGE.FREQ	-----	122
APPENDIX AA	ROUTINE EW.ROUTINE	-----	123
APPENDIX BB	EVENT NO.CONTACT	-----	124
APPENDIX CC	ROUTINE SIEZE.NET	-----	125
APPENDIX DD	EVENT END.XSMN	-----	127
APPENDIX EE	ROUTINES CO.MSG, BN.MSG, BDE.MSG, and DIV.MSG	-----	129
APPENDIX FF	EVENT DUMP.MAILBAG	-----	130
APPENDIX GG	ROUTINE ABORT.MSG	-----	131
APPENDIX HH	EVENT MSG.GEN	-----	132
APPENDIX II	COMMUNICATIONS OUTPUT EXAMPLE	-----	134
BIBLIOGRAPHY	-----	-----	144
DISTRIBUTION LIST	-----	-----	145



## LIST OF FIGURES

FIGURE 1	STAR DECISION LOGIC	-----	15
FIGURE 2	COMMUNICATIONS ENTITY AND POINTER DIAGRAM	--	26
FIGURE 3	ROUTINE DECISION	-----	29
FIGURE 4	ROUTINE DEC.CO	-----	30
FIGURE 5	ROUTINE DEC.BN	-----	31
FIGURE 6	ROUTINE DEC.BDE	-----	33
FIGURE 7	ROUTINE DEC.DIV	-----	35
FIGURE 8	EVENT COMMO. ATTEMPT	-----	42
FIGURE 9	EVENT END.XSMN	-----	46
FIGURE 10	ARRAY TABLE DIAGRAM	-----	63
FIGURE 11	ARRAY BNCORD DIAGRAM	-----	64
FIGURE 12	ARRAY COCORD DIAGRAM	-----	65
FIGURE 13	ARRAY BDECORD DIAGRAM	-----	81
FIGURE 14	ARRAY BNNTBL DIAGRAM	-----	82
FIGURE 15	ARRAY DIVCORD DIAGRAM	-----	84
FIGURE 16	ARRAY RRPOINT DIAGRAM	-----	85



#### ACKNOWLEDGEMENT

This implementation of communications for the movement decision logic in the STAR Combat Model utilizes a version of the communications model initially developed by Cpt. William Haislip and subsequently modified and revised by Associate Professor Arthur Schoenstadt.

The extensive computer resources required during development and testing of the model were ably provided by the staff of the W. R. Church Computer Center.

I also wish to express my heartfelt thanks to my wife, Linda, without whose dedicated support and nimble fingers this thesis probably would never have gotten through the typing phase.



## I. INTRODUCTION

In order to accomplish its mission of winning the land battle, the Army must perform three basic functions: shooting, moving, and communicating. Therefore, it is reasonable to say that any combined arms combat model must portray these three functions with a commensurate degree of accuracy in order to realistically represent the combat process.

The question of precisely how to model communications between the elements, units, and individuals represented in a combined arms combat model is a critical one for the model designer. Communication must be an integral part of virtually every conceptualization of combat more complex than a one-on-one duel. As a result, communication is one of the most significant areas which the combat modeler must consider and with which he must deal effectively.

The assumptions made about the amount of communications allowed between elements, the time required for the communications to be completed, and the effect of communications (or lack thereof) frequently can be so far-reaching that they completely dominate many other effects which the modeler would like to study. They may also cause the model to be rejected as invalid. For this reason, the communications-related assumptions must be carefully thought out and implemented if the combat model is to be realistic and useful.

The original efforts in developing the STAR Combat Model were directed towards producing a high resolution Monte Carlo simulation of modern combined arms warfare. It was decided to base the terrain model on the work done by Needels for his thesis research in March 1976. He



demonstrated that bivariate normal distributions could be used to give a very good functional representation of continuous terrain. The STAR concept was to develop an event step simulation in which the impact point of each direct fire projectile was computed and every firepower activity of the represented elements was simulated insofar as possible. The movement of units, firing of weapons, and all of the related functions such as target acquisition, target selection, attrition, tactics and so on were modeled in considerable detail. Because the model was to focus on the evaluation of weapons and tactics, the original studies set aside the problem of communications for future study in order to get on with the development of the "shooting and movement" portions of the model. A description of the routines, events, and arrays of the STAR model which impact on the development of this thesis is provided in Appendix A.

The original work on the STAR model was done by Hagewood and Wallace in December 1978. At that time and stage of development the model portrayed a blue company under attack by a red battalion. Artillery and close air support/air defense modules were under development but were not included in the model at that time. A dynamic route selection module which considered enemy locations, terrain, and the tactical situation was being considered for inclusion in the model as was an improved module for target selection. In addition, fuel and ammunition resupply modules and a communications module were under development.

In March of 1977, four theses covering those areas mentioned above were written. The first, by Starner, developed a two-sided field artillery stochastic simulation which played both red and blue artillery at the battalion level. Provisions were made for upgrading the model to portray artillery at the brigade level when the brigade level was included in the STAR baseline.



The second, by Kramer, developed a deterministic simulation model for dynamic tactical route selection based upon the route selection technique used in the DYNTACS combat simulation model. Kramer's model was written in FORTRAN, and it was planned for eventual incorporation into the STAR model.

The third, by Broussard, developed a dynamic model for the tank commanders' target selection process based on eleven parameters fit by a regression model using responses from 128 tank gunners.

Finally, the fourth, by Haislip, developed a stand-alone communications model which will be discussed in a later paragraph.

In September of 1979, Caldwell and Meiers developed an air-to-ground and ground-to-air combined arms simulation designed for inclusion in the STAR model. At this time, STAR had been upgraded to include representation of a blue battalion under attack by a red regimental force; however, the production version which was used for several studies for TRADOC still did not include any of the modules such as the field artillery, dynamic route selection, dynamic target selection, or communications.

In his thesis work in March 1979, Haislip developed a model for single channel FM communications at the company and battalion level. Despite the fact that this code was written with the idea of incorporating it into STAR, there were several shortcomings which precluded its inclusion. First and most significant, the code was task specific so that in order to use it for the wide range of applications represented in the STAR model much of the logic and many of the variable and attribute names would have had to be altered. Second, the communications model only portrayed the communications for internal platoon fire coordination and field artillery fire direction. It was subsequently decided



that, for the near term, STAR would not explicitly model the communications below the company level for reasons which will be outlined in Chapter V. In spite of these shortcomings, the model written by Haislip did provide valuable insights into how to approach the problem of user-transparent communications modules for the higher echelons.

Since the purpose of this thesis was to design and implement a communications model for the movement decision logic within the framework of the existing STAR Combat Model, Haislip's work was used as the basis for the communications implementation with modifications and extensions in order to make the communications interface as robust as possible within the constraints imposed by the existing STAR organization. A description of the routines and events of the communications model is provided in Appendix B.

Because of the evolutionary nature of the STAR model there are always several "current" versions of the model in some stage of work. For the remainder of this thesis, any reference to STAR is a reference to the current production version of the model which does not include those modules under development. Also, references to the communications model refer to that module which was developed as a stand-alone model of the communications process using the decision logic flow from the production version of STAR. Hopefully, this communications module will be enhanced and included in one of the STAR baseline production models scheduled for completion later this year.



## II. DESCRIPTION OF THE STAR MODEL

The STAR combined arms model is being developed at the Naval Postgraduate School to investigate the effectiveness of both tactical doctrine and materiel in a mid-intensity combined arms conflict. As previously noted, the model uses functionally generated continuous terrain and currently simulates a blue battalion under attack by an armor heavy red regimental force. The terrain currently in use simulates a 10 kilometer by 10 kilometer area near Fulda, Federal Republic of Germany; however, other terrain boxes have been developed including one 40 kilometers by 80 kilometers which is planned to be used with simulation of a blue brigade-level force under attack by a red division-level force.

The current version of STAR simulates individual weapons systems such as XM1 tanks, TOW's, BMP's, T72 tanks, and dragons. These entities detect enemy systems, select and acquire targets, load and fire their weapons, and move about the simulated battlefield based upon input parameters supplied by the model user. The user defines the types of weapon systems, the ammunition available, the tactics to be used for selecting and engaging targets, and the force ratio and range breakpoints used to determine if a unit wants to withdraw from its current position. A line-of-sight module is used to determine whether or not one system can see and, therefore, engage an enemy system based upon their respective locations on the continuous terrain.

Air defense, dismounted infantry, and artillery modules are under development for inclusion in the model. Also, preliminary work has been done on a vehicle evacuation and repair module to simulate one of the major logistics



functions.

During the initialization process for the STAR model, each company is assigned a weight based upon the users' view of the importance of the particular company in the overall defensive posture of the battalion. This value, COWT, is an attribute of the company for a particular phase line and is used by the movement decision logic. A company can and frequently does have different values of COWT for different phase lines. Also during the initialization, the user assigns a value to column 4 of the array TABLE for each company and weapon system on every phase line. This value represents the restriction status of the unit and must be a one or a zero. If the value is a one, the company or weapon system is restricted from moving from that phase line position unless given permission by its superior commander. If the value is zero, the company or weapon system can move at will. TABLE also stores codes for all of the tactical actions allowed for each company and weapon system on each phase line. Associated with each of these action codes, the TABLE array contains a force ratio and an attrition level which trigger the action to be taken. (Technical Report NPS55-79-023 by Parry and Kelleher contains a complete description of the parameters stored in array TABLE.)

The movement logic in STAR is handled by Routine DECISION which uses the unit number of the unit requesting or being ordered to move, the row number in the TABLE array of the weapon system to be moved by the unit, and the appropriate column number in the TABLE array. This information is combined with whether or not the weapon system is restricted from moving from its present phase line position, which is in column 4 of the TABLE array. A flow diagram of the logic in Routine DECISION is given in Figure 1. The arrays used by the movement decision logic are described in Appendix A.



STAR DECISION LOGIC

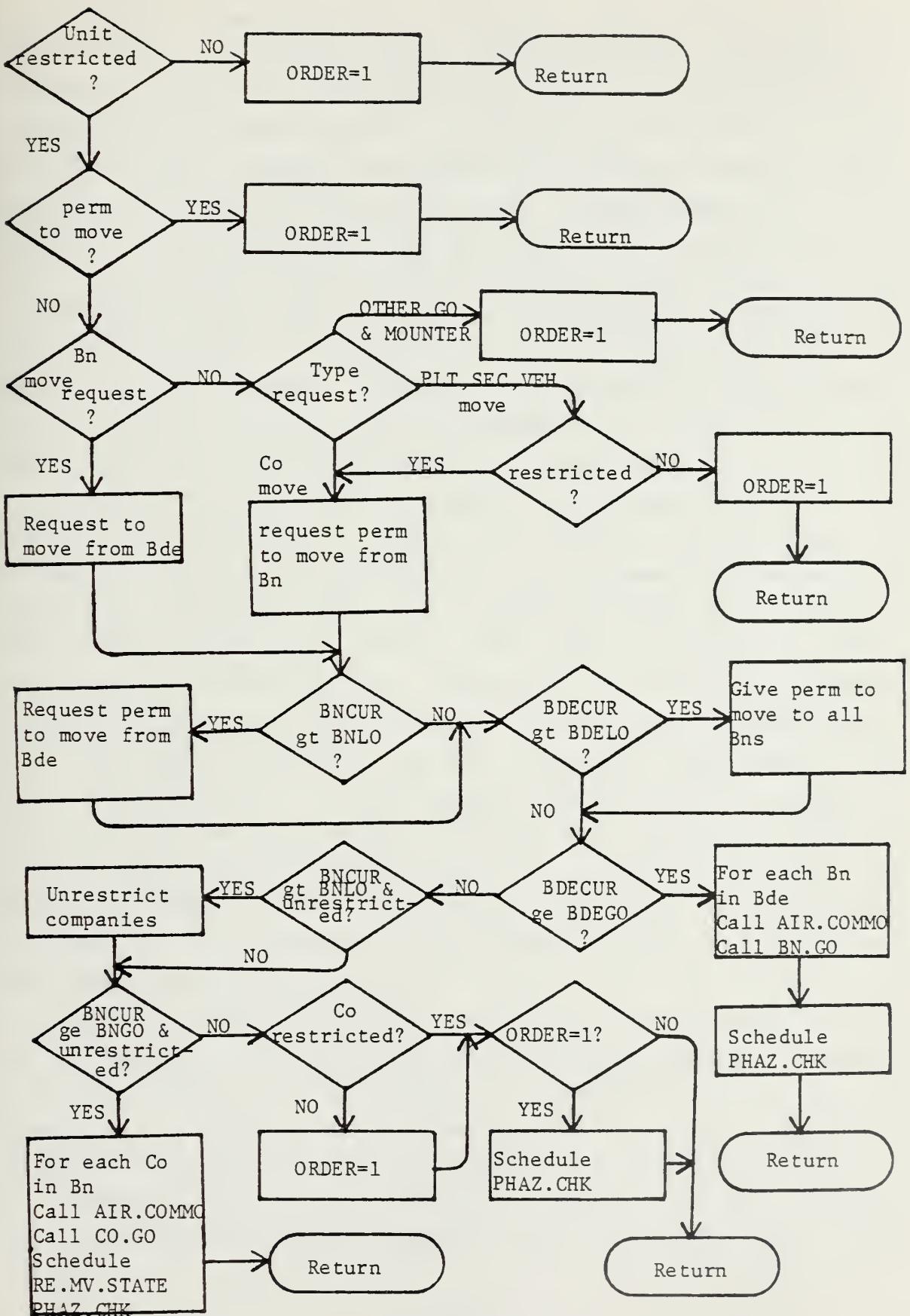


FIGURE 1



During simulation of a battle, blue forces may be allowed by the model users' choice of input parameters to move out of their current position as a result of heavy attrition by enemy forces or because of the range to the nearest enemy element. Each shot at a blue element causes routine BUG.CHK to check the range to the enemy. If it is less than the value supplied by the user in column three of the array TABLE for that phase line and that weapon system or company, then routine DECISION is called to determine if the weapon system or company is restricted or is allowed to move from its current position. Similarly, routine ACTION calls routine DECISION as blue systems are killed by enemy fire and force ratio decision points are reached.

In order to determine when a battalion will give its companies permission to move and when it will order its companies to move, each battalion has three attributes. Two of these, BNLO and BNGO, are assigned values by the user. The third, BNCUR, is set to zero when a battalion first occupies a phase line. Whenever a company requests permission to move, its weight is added to the BNCUR value. If this value exceeds the user defined BNLO value, the battalion gives all of its companies permission to move at will. If the value of BNCUR exceeds the user input value of BNGO, the battalion orders all of its companies to move. As long as BNCUR is less than BNLO, restricted companies and systems of that battalion stay restricted and are not given permission to move.

When routine DECISION is called, the restriction status of the company or weapon system in array TABLE is checked, and the BNCUR value is updated if required. Then, if the company or system is allowed to move, the appropriate movement routine is called to handle the movement process.

Since in the current version of STAR all of the checking of the company status, updating of the BNCUR value, and



calling of the movement routines is done in the same routine, an assumption is made of perfect and instantaneous communications from the company to the battalion and back to the company. This and the other assumptions of STAR dealing with the communications and decision processes are discussed in the next chapter.



### III. COMMUNICATIONS ASSUMPTIONS OF STAR

The current STAR Combat Model makes several far-reaching assumptions concerning communications during simulated battles. These assumptions are driven by the fact that neither communications nor electronic warfare is modeled explicitly in the current production version of STAR. As noted in Chapter I, the modeling of these phenomena were left for later work by the original studies done on STAR. In order of decreasing importance the implicit assumptions made in the STAR model concerning communications and electronic warfare are listed and explained below.

- 1) Communications of information from one element to another takes no time. This assumption means that a primary effect of communications on the battle process, the time it takes to reach decisions and affect the actions or posture of remote units, has been ignored.
- 2) Communication between units is independent of the locations of the units and of the intervening terrain. In practice, the intervening terrain often precludes any communication between units.
- 3) All designated recipients of a communication receive exactly the same information at exactly the same time. This means that messages are never lost and all are handled with perfect efficiency.
- 4) The probability of successful communications between two elements is not affected by the combat situation nor by the kill condition of either element. This means that communications gear is never damaged during the battle to such an extent that it cannot be repaired by the operator, or replaced.



- 5) All methods of communications are equivalent since they all take no time and they all have the same probability of success.
- 6) All commanders are always instantaneously aware of a change in the status of any of their units.

Of these six assumptions, the one of paramount importance is the first because of its great effect on the outcome of the battle. This thesis focuses on implementing a communications model for the movement decision logic of the STAR model and on providing a method of modeling messages so that this first assumption can be eliminated in favor of one which deals only with the distribution of the time delays caused by transmission of messages from one entity to another.

A description of the problems caused by these assumptions is presented in the next chapter.



#### **IV. DESCRIPTION OF THE PROBLEM**

Because of the intense effort to implement a production version of STAR as an effective combined arms combat model, the development of the communications aspects of the model has not been the primary concern of the model developers. The predominant view concerning communications between units has been that the oversimplified portrayal would be adequate until such time as the model was sufficiently sophisticated to allow the communications to play a significant role in the exercise of the model.

The lack of an adequate communications model as outlined above together with the assumptions of the STAR model regarding communications which are stated in Chapter III have a great effect on the results of the model in certain critical situations. For example, since the loss of time in transmitting information is ignored, the battalion commander is instantly aware when one of his companies is being heavily attrited by the enemy and wants to withdraw from a particular position, and no time is taken up by the decision process which generates the order or the communication process which transmits the movement authorization to the company. This, in turn, means that this particular blue company is, on the average, attrited less than it would be if the company had to wait some length of time to get permission to move from its current position. Thus blue casualties are lower than they would be for a more realistic simulation of the decision and communications process.

This thesis work was begun at a time when it was decided that a more detailed treatment of communications was necessary. It was determined that this was an area of the



model where significant improvements in model realism could be made. At the same time, it was apparent early in the development of the communications model that major portions of the STAR code would have to be completely rewritten to implement a realistic communications simulation model. Many of the decisions which are based upon communicated information are "hard-wired" into the STAR code which means that in order to change these decisions, the code itself had to be radically altered to provide for more flexibility. As noted previously, this thesis work was begun with the goal of implementing a communications model within the STAR model which would handle the message flow for the movement decision logic. This model was to be sufficiently flexible and expandable to handle the communications requirements for other functions as they are converted to the explicit communications representation. A description of the assumptions of the communications model is provided in the next chapter.



## V. ASSUMPTIONS OF THE COMMUNICATIONS MODULE

The communications model developed in this thesis makes the following assumptions.

- 1) A message on the communications net will not be interrupted by another message trying to get on the net. This means that there is no scheme for handling different message precedence levels.
- 2) Human error in interpretation of the messages is negligible.
- 3) Signal strength during transmission of a specific message does not vary significantly.
- 4) The probability of jamming a given message is independent of whether or not a previous message on the same net was jammed.
- 5) None of the unit commanders are killed during the course of the battle. This is accomplished by setting all the commanders ammo counts to zero and by setting each commander's ACVT.TYPE attribute equal to one.
- 6) Below the company level, communications by messenger takes approximately the same time as electrical communications and therefore communications at this level is assumed and is not modeled.

Some brief comments on these assumptions are in order and they are provided below.

Until such time as the communications model is sufficiently sophisticated to allow for modeling of traffic priority, the first assumption provides an adequate model for a process where all traffic is of the same relative importance.

The second assumption covers a multitude of faults which



exist in the real world communication process. However, just as it is not feasible to try to model the differences in viewpoint between the red and blue commanders except in terms of doctrine, it is not feasible to try to model the possible ways a message can be misinterpreted since this is a function of so many disparate factors.

The third assumption provides a good approximation to reality except during bad weather when atmospheric conditions change with great rapidity, and when jamming is encountered. This assumption has an explicit proviso, that is that the message lengths must be short relative to the time scale for atmospheric or equipment changes.

The fourth assumption is applicable to this version of the communications model but will undoubtedly be superceeded by later revisions which are anticipated to model the phenomenon of jamming in great detail. In the current version the probability of jamming is set by the user by his choice of the values for the global variables which are used by the SIMSCRIPT random number generator. A description of the global variables used in the communications model is provided in Appendix C.

The fifth assumption is made because this entire area of battlefield synergism is not being played in the model. In real combat, commanders are killed and units lose or gain effectiveness based upon the capabilities of the subordinates who assume command. There is no data to support any sort of model of this gain/loss of effectiveness, and it is not anticipated that any attempt will be made to incorporate this sort of detail in the model since it turns primarily on capabilities of individuals such as leadership, judgment, and decisiveness which are essentially not quantifiable.

The rationale for the sixth assumption and for not modeling the communications below the company level is that



these emitters are generally so close together that it is almost always possible to pass traffic between them, and jamming them becomes prohibitively expensive in terms of level of effort and expenditure of resources. Also, because they are relatively close together, critical information can be passed by messenger in a time that is roughly equivalent to the time required to pass the information by electronic methods. This is not true for messages between the higher level commanders where the time for a messenger to deliver the information would be orders of magnitude larger than the time to deliver the information by electronic means. In addition, modeling all these details would consume valuable computer resources which can be more profitably used to represent other aspects of the combined arms battle.

Unlike the communications model currently employed by the STAR model which assumes no loss of time in transmitting information from one unit to another during the course of the battle, the one developed in this thesis provides the user of the model with almost complete freedom in his choice of how to model the distributions of the time delays involved in the communications and decision processes. The guidelines for model parameterization are covered in detail in Chapter VII. A complete description of the communications model is provided in the next chapter.



## VI. DESCRIPTION OF THE COMMUNICATIONS MODULE

The communications model developed in this thesis is an amalgum of ideas and concepts from many sources. The logic of the STAR model necessarily makes many decisions which in real life are based upon information communicated to the commanders by sources outside their immediate area. Most of this information is provided to the commander in the form of a written or verbal message. While these messages do not conform to any set format, there are certain minimum essential elements of information contained in them which are determined by the type or intent of the message. For example, as a minimum, a request by a company for permission to move from a specified position must include a unit designator and some type of code to identify this message as a request to move. Since different types of messages will necessarily have different minimum information contents, it was decided that the messages and their text portions should be modeled separately. In order to help the reader with understanding the logic flow which will be discussed in this chapter, the entities and their pointers which are used by the communications logic are diagrammed in Figure 2. Also, the arrays and sets used by the communications logic are described in Appendices F and G respectively.

To represent the flow of information within the model, messages are generated as temporary entities whenever a decision point is reached where information must be passed between two subscribers which are spatially separated or are not from the same command level. There is one exception to this policy: the internal company communications. Communications from individuals to their platoon leaders and from the platoon leaders to the company commanders are



COMMUNICATIONS ENTITY AND POINTER DIAGRAM

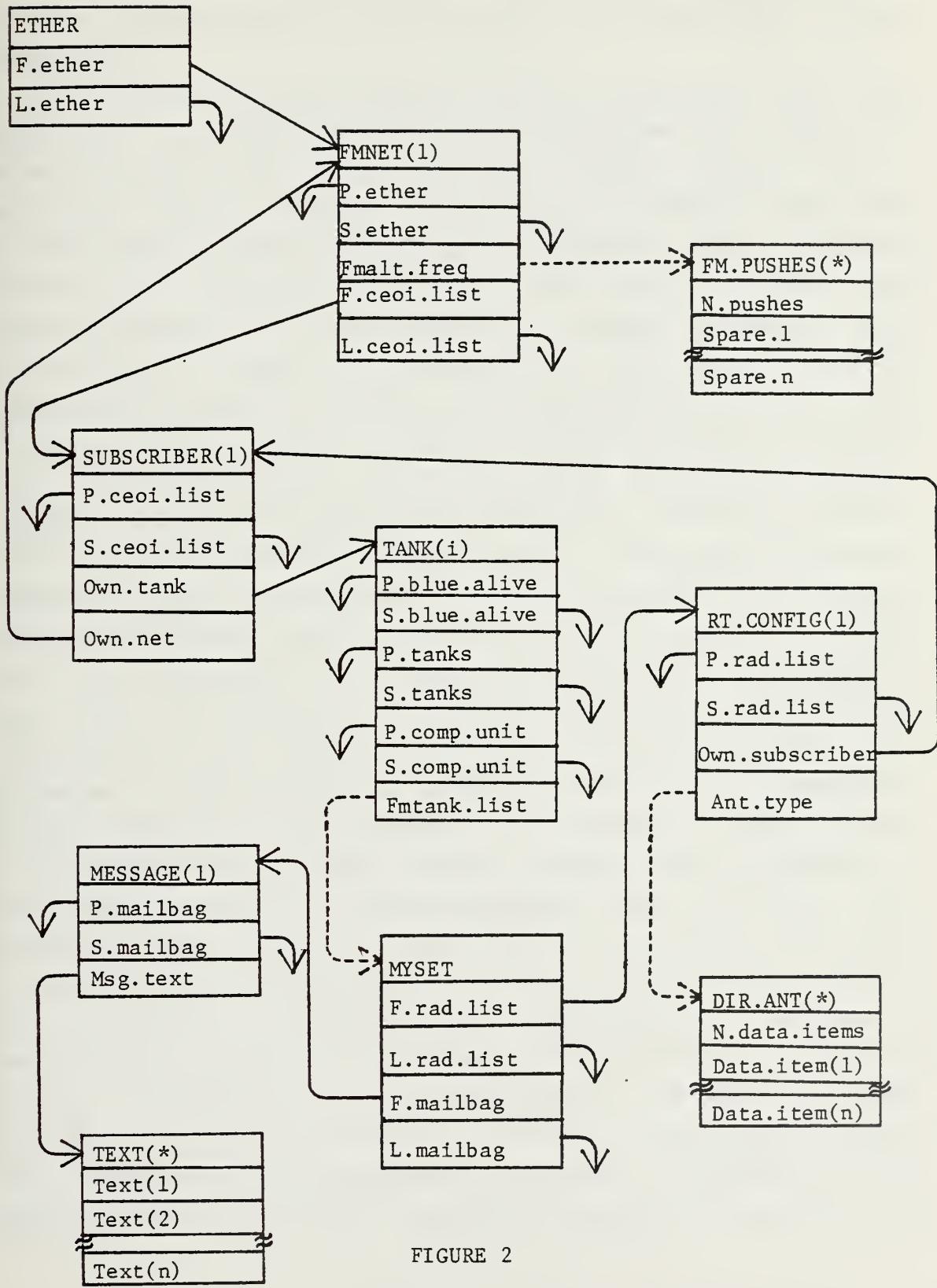


FIGURE 2



assumed to be 100% reliable and take no time. This is not the case for communications between the company commanders or for communications between any of the higher level commanders.

The communications module being implemented uses the same basic decision logic as that diagrammed in Figure 1. However, since the primary emphasis of this thesis was to model the communications delays and the effects they have on the eventual outcome of the simulation, the DECISION routine had to be completely rewritten to allow the representation of these delays. Routine DECISION was actually taken apart and broken down into seven mutually disjoint new routines. These seven routines utilize the same arrays and sets as the original DECISION routine to model the decision process. Four of these routines, DEC.CO, DEC.BN, DEC.BDE, and DEC.DIV, handle the movement decision logic at the various command levels. The fifth routine, DECISION, acts as the driver which initiates the movement decision process when called by either BUG.CHK or ACTION from the STAR logic. The sixth routine, EXIT, provides an exit from the movement decision logic under specified conditions. The seventh routine, GEN.MOVE.DECISION.MSG, produces the appropriate message which is transferred between these routines carrying the information required for the decision to be made. These routines are discussed in more detail later in this chapter along with all the communications routines and events necessary for the transfer of messages between them. The SIMSCRIPT code for these modules is provided in Appendices H through O respectively. Instead of changing the information known by all the commanders at all levels when a company commander decides to request permission to move, the communications model stochastically generates and sends a MESSAGE from the company commander to the battalion commander containing the



request. When this MESSAGE transmission ends, the model provides the STAR decision logic with the information necessary for a decision by the battalion commander and then stochastically models the transmission of the decision by generating and sending a MESSAGE from the battalion commander to the company commander. If the users choice of inputs does not allow the battalion commander to give permission to move to the company commander, another MESSAGE is generated and sent. This new MESSAGE goes to the brigade level and when the transmission ends, the decision logic at the brigade level either generates the decision and sends a MESSAGE back to the battalion or passes a further request for permission to move on to the division level. As the model stands, the division level cannot send a message higher and thus is the ultimate decision authority. The user should be aware that his choices of break point bounds and restriction status for the simulated units are critical for a realistic simulation. Flow diagrams of the modified decision logic used in the communications model are given in Figures 3, 4, 5, 6, and 7. In addition, a partial listing of the output of the model is provided as Appendix II. This listing has been greatly shortened, but serves to illustrate the type of output which can be expected from the model during execution. A description of the flow of the communications model routines and events will be given next to familiarize the reader with the details of the model.

#### A. COM.MAIN

The SIMSCRIPT code for routine COM.MAIN is provided in Appendix 3. It is the first of the communications model routines to be executed. It is called from BL.CREATE and it in turn calls all the initialization routines for the



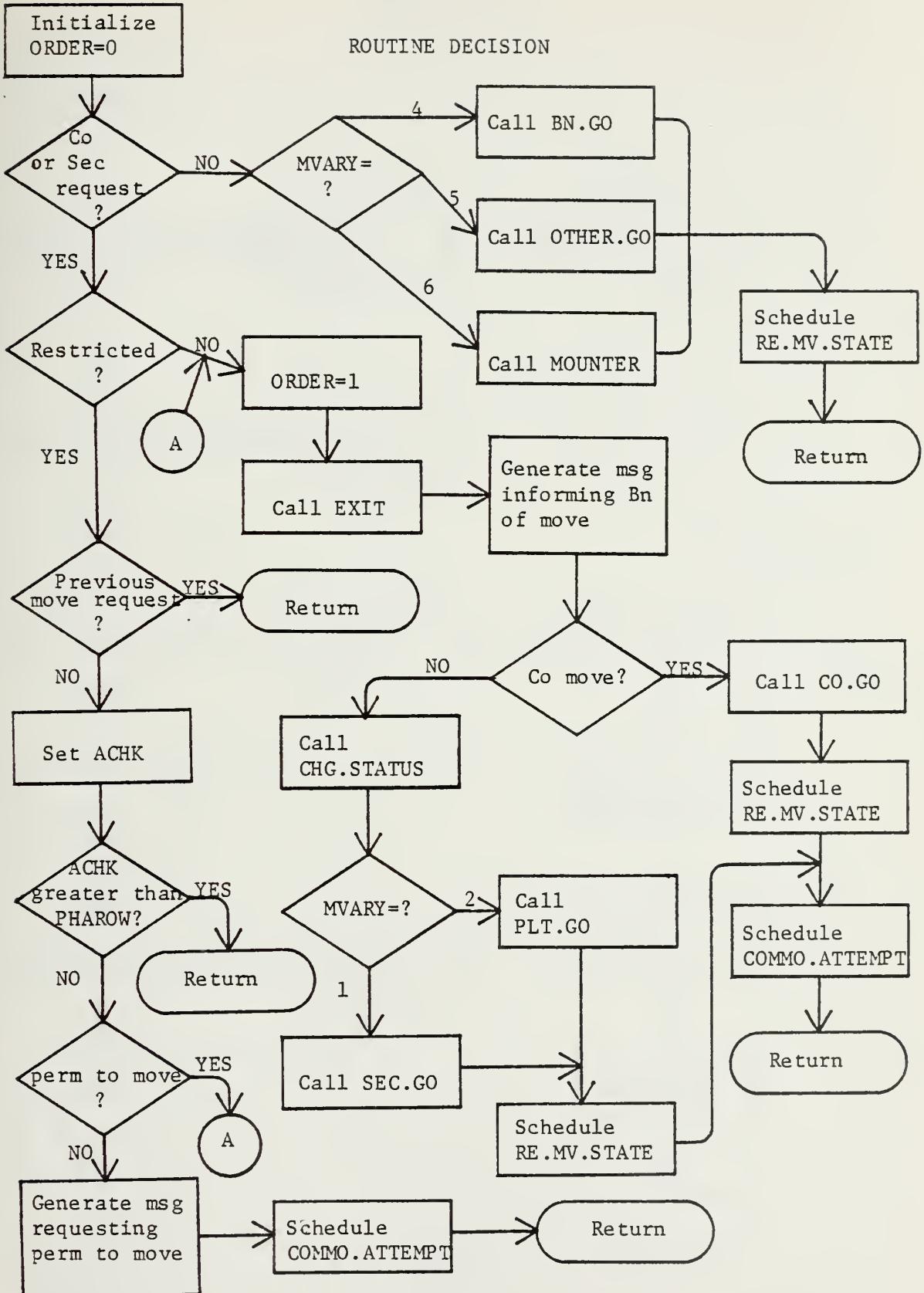


FIGURE 3



ROUTINE DEC.CO

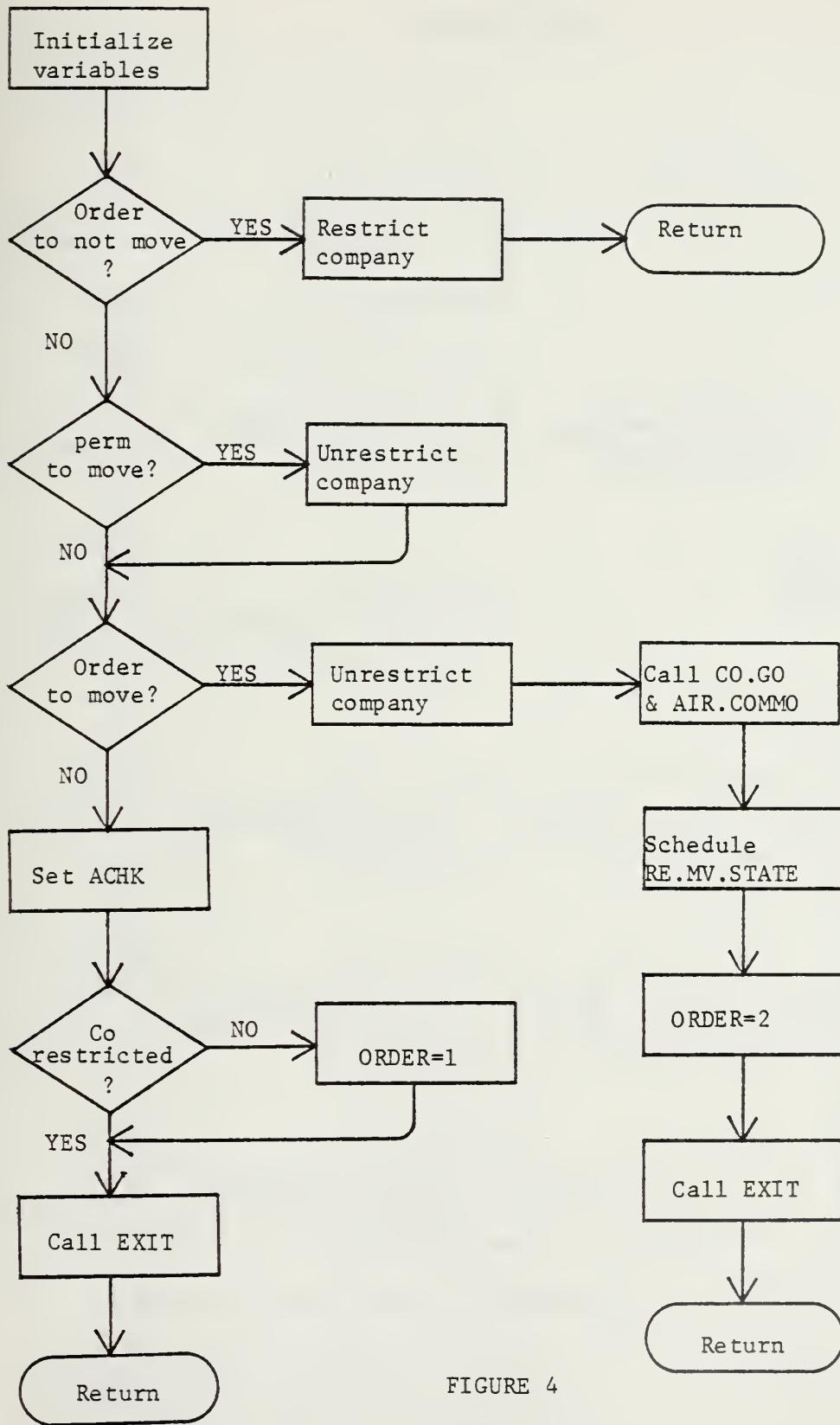


FIGURE 4



ROUTINE DEC.BN

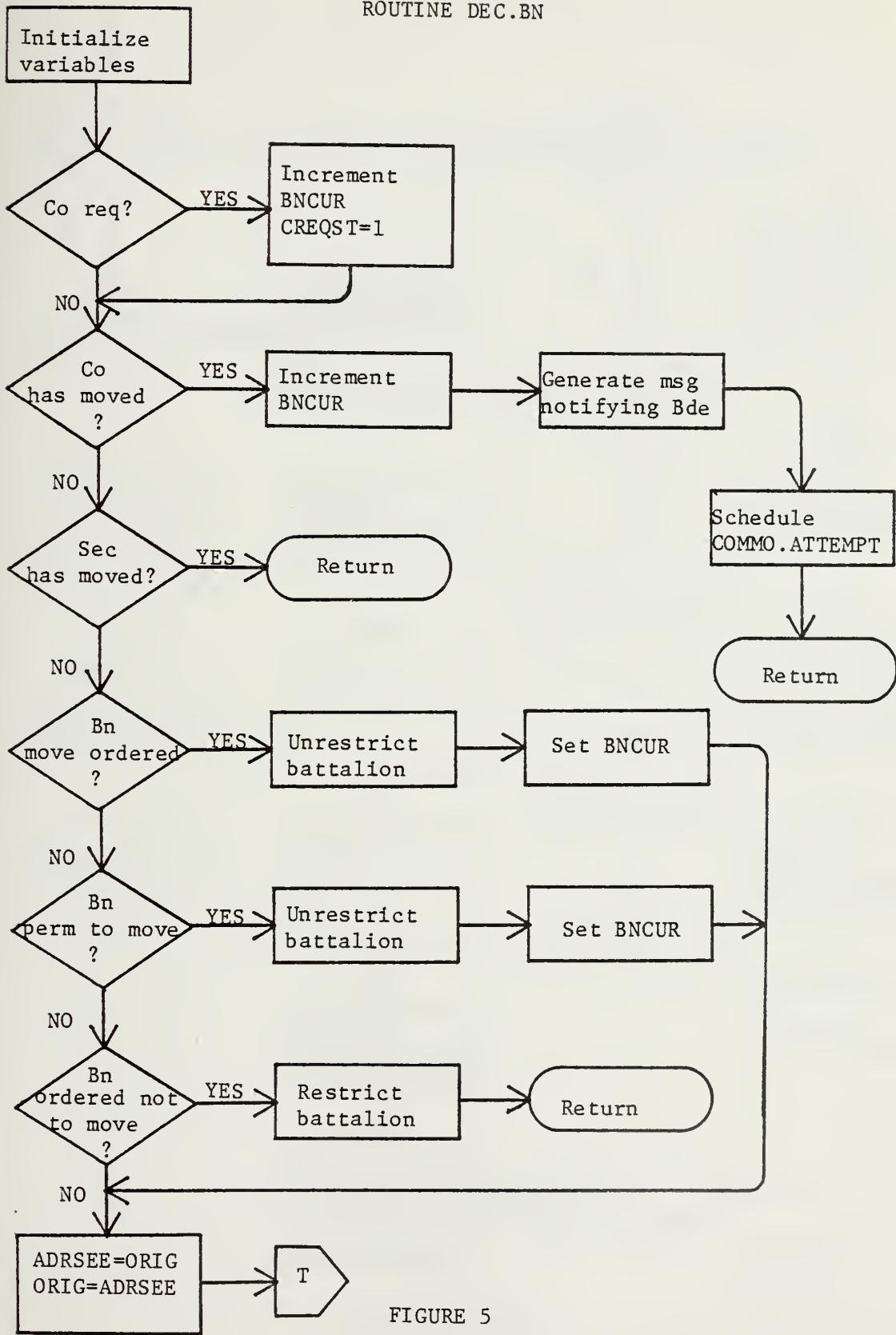


FIGURE 5



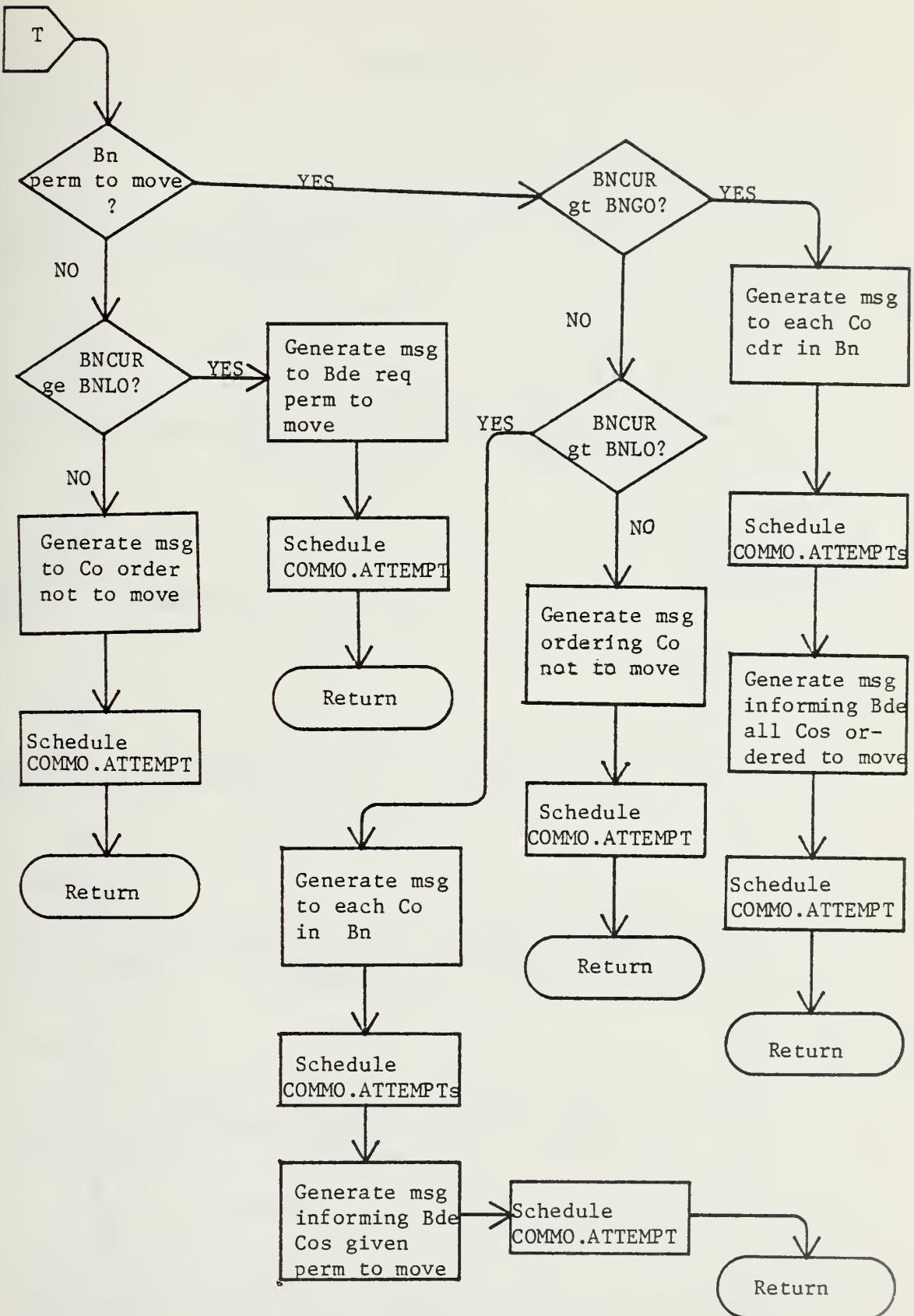


FIGURE 5 (cont.)



ROUTINE DEC.BDE

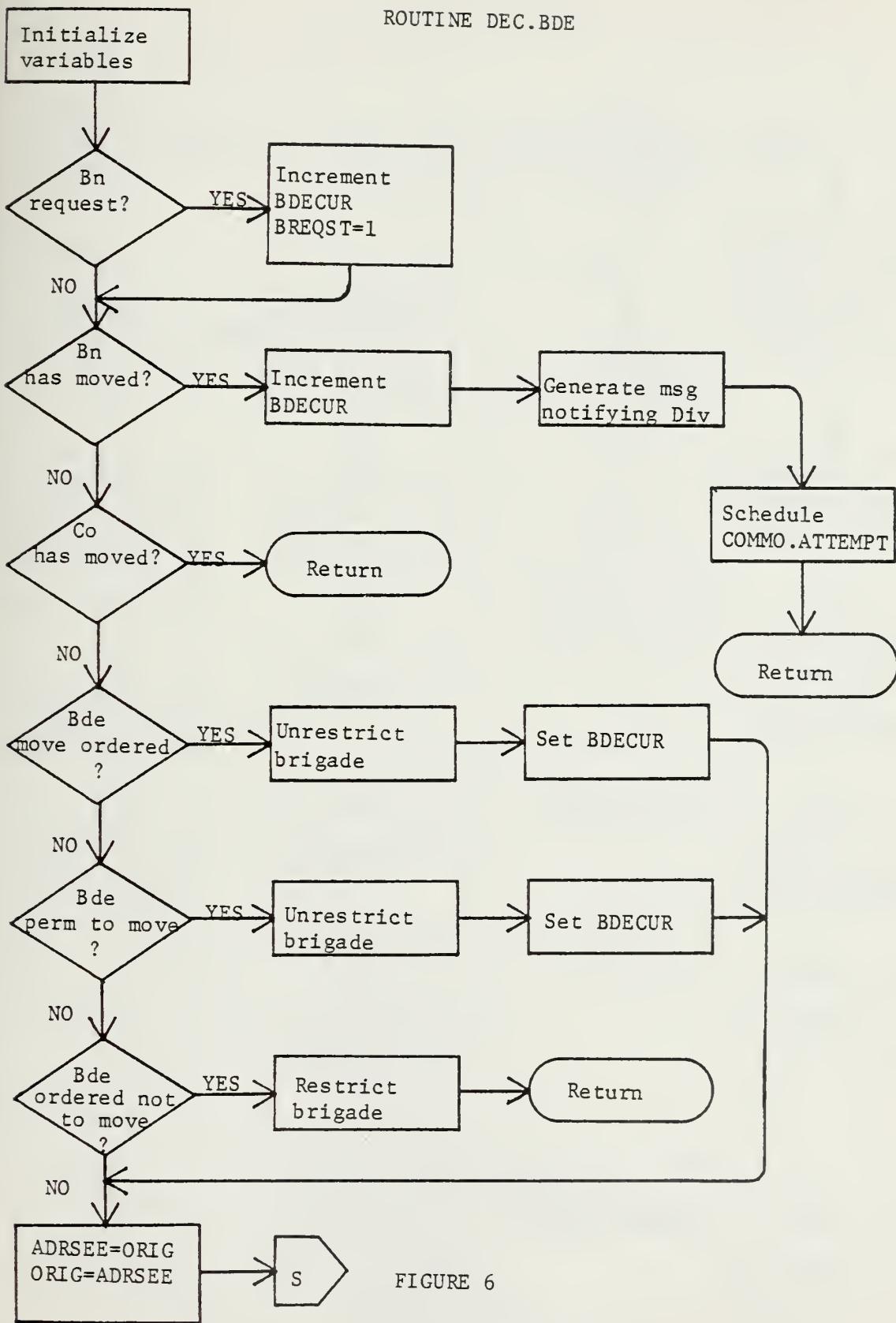


FIGURE 6



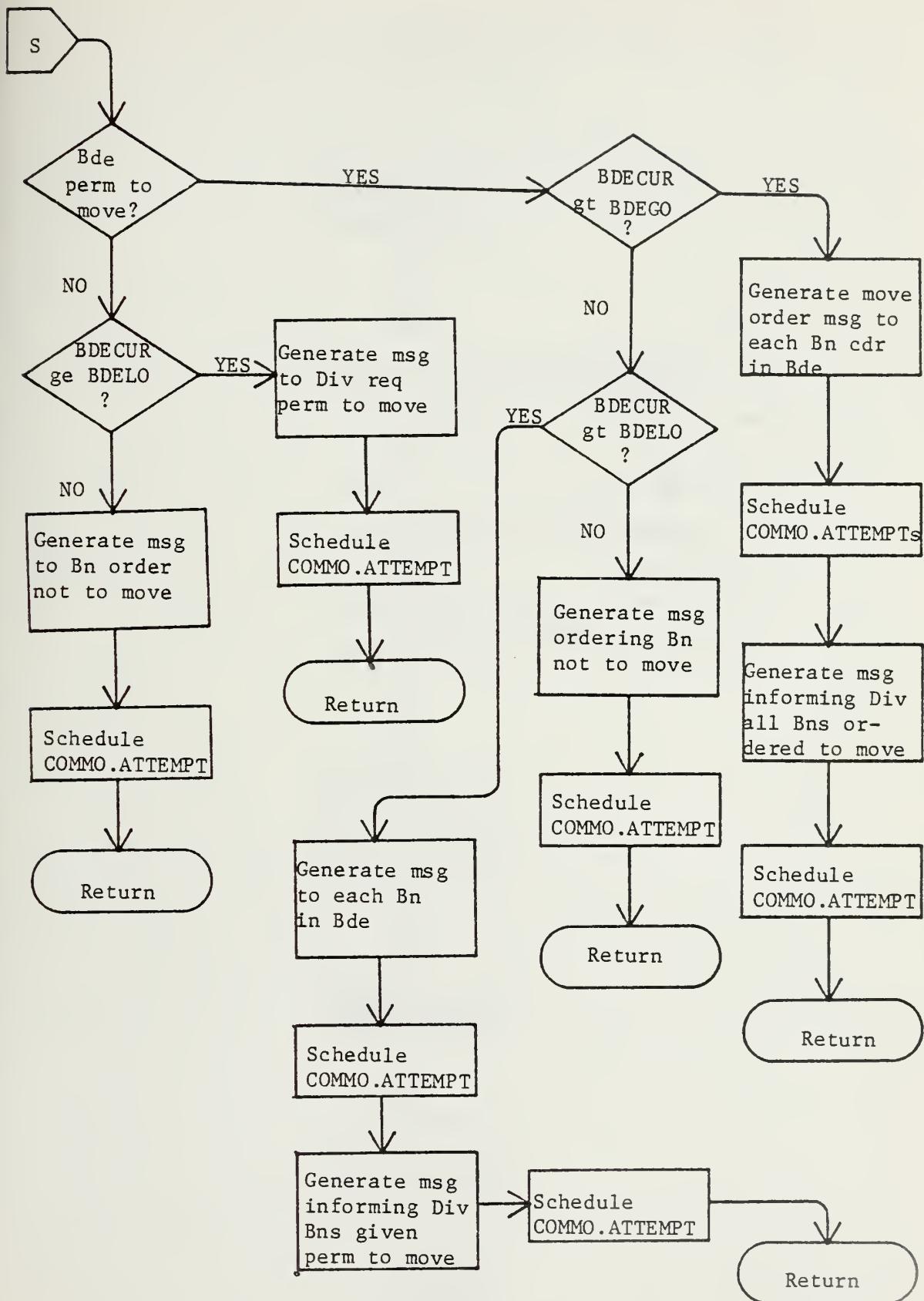


FIGURE 6 (cont.)



ROUTINE DEC.DIV

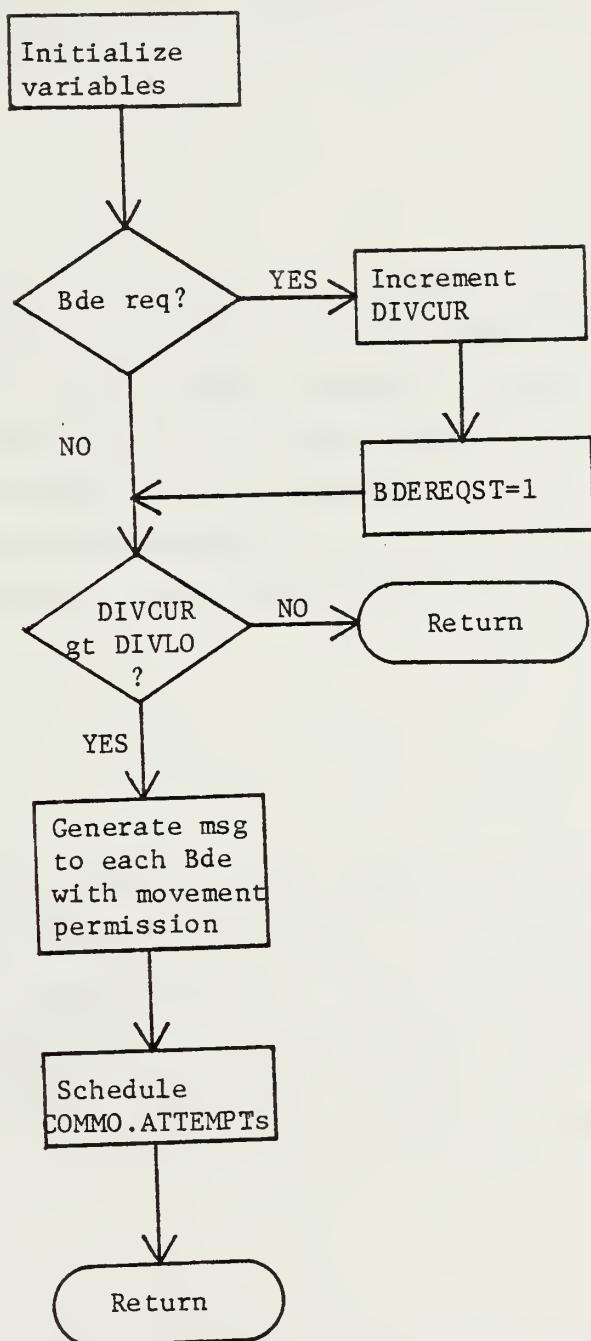


FIGURE 7



communications module before listing the attributes of each B.MVR.CDR, COMPANY.COMMANDER, BN.COMMANDER, BDE.COMMANDER, DIV.COMMANDER, and UNIT. COM.MAIN is the last executable routine called by BL.CREATE, and when it terminates, the SIMSCRIPT timing routine takes control since the simulation has already been started.

## B. BUILDNETS

The SIMSCRIPT code for routine BUILDNETS is provided in Appendix P. It reads in the data on the FMNETs which are to be simulated and on which UNITS belong to each FMNET. It then creates the nets and the SUBSCRIBERS and places the SUBSCRIBERS in the CEOI.LIST for their net. It also reads any alternate frequencies which are assigned to the FMNET and stores them in the array FM.PUSHES. It then returns control to COM.MAIN.

## C. CHECKNETS

The SIMSCRIPT code for routine CHECKNETS is provided in Appendix Q. It goes through each FMNET that has been created and prints out the information on them. It prints the net number, the net frequency, the number and frequency of any alternate frequencies, and the number of SUBSCRIBERS assigned to the net before returning control to COM.MAIN.

## D. BNNTBL

The SIMSCRIPT code for routine BNNTBL is provided in Appendix R. It cycles through all the FMNETs and creates and



prints out two tables. The first, NETTBL, is used during execution of the communications routine. It contains the identity numbers of the SUBSCRIBERS belonging to the ith FMNET in the ith row. The second table has the address pointers to the SUBSCRIBERS in the ith FMNET in its ith row. After printing both tables, BNETTBL returns control to COM.MAIN.

#### E. ROUTINE COMMGEAR

The SIMSCRIPT code for routine COMMGEAR is provided in Appendix S. It reads in the data on how many UNITS/TANKS have communications equipment and creates a MYSET for each of them. It then creates the RT.CONFIGs for each TANK and reads in the characteristic data for each RT.CONFIG assigned to each TANK and files the information as attributes of the RT.CONFIG. If the data indicates that the antenna is directional or remoted, a message is printed and the job terminates since there are no provisions in the current model for handling directional or remoted antennas. Eventually, these are planned for inclusion, however. The routine then cycles through the FMNETs and sets the first message number for each SUBSCRIBER in each net equal to n001, where n is the identity number of the SUBSCRIBER. Finally, this routine places the SUBSCRIBERS in the nets and returns control to COM.MAIN.

#### F. ROUTINE COMMCHECK

The SIMSCRIPT code for routine COMMCHECK is provided in Appendix T. It cycles through each TANK in BLUE.ALIVE and prints out the communications gear assigned to each TANK.



For each RT.CONFIG it prints out the x,y, and z coordinates, the type of radio, the radio mode of operation, the antenna type, and the location of the antenna which currently must be the same as the location of the TANK. If either a directional or remoted antenna is detected, a message is printed and the program halts. This is done because the routines which are planned to handle this have not yet been included in the model. Finally, it prints the number of the net to which each RT.CONFIG is tuned before returning control to COM.MAIN.

#### G. ROUTINE INITIALIZE

The SIMSCRIPT code for routine INITIALIZE is provided in Appendix U. It reads in the values for the parameters used by the communications routines for generating random numbers and for controlling the print out of data. It also reads in the lower and upper bounds for the destination codes of messages being sent to the company, battalion, brigade, and division levels. It also calls routines STAT.DUMP and SSTAT.DUMP before it returns control to COM.MAIN.

#### H. ROUTINE STAT.DUMP

The SIMSCRIPT code for routine STAT.DUMP is provided in Appendix V. It lists the attributes of each COMPANY.COMMANDER, BN.COMMANDER, BDE.COMMANDER, and DIV.COMMANDER and is used primarily as an aid to debugging. In addition to the initial call from routine INITIALIZE, it is called from routine END.XSMN periodically based upon the value supplied by the user for global variable COM.DATA.DUMP. Complete descriptions of the global variables



used in the communications model are provided in Appendix C while the attributes of the permanent and temporary entities are described in Appendices D and E respectively.

## I. ROUTINE SSTAT.DUMP

The SIMSCRIPT code for routine SSTAT.DUMP is provided in Appendix W. It prints out the contents of the BDECORD, BNCORD, and COCORD arrays. This is another of the print routines designed to aid in debugging the code if it should fail. The arrays used in the communications model are listed and explained in Appendix F.

Once initialization of the modules has been accomplished and the SIMSCRIPT II.5 timing routine has taken control of the system, the event step sequence determines the order in which the routines and events will take place. The following description of the routines and events is done as if tracing the flow of a specific decision through the system. Whenever in the following narrative the phrase "a message is sent" is encountered, it should be taken to mean "a COMMO.ATTEMPT is scheduled."

## J. ROUTINE DECISION

The SIMSCRIPT code for routine DECISION is provided in Appendix H. The flow diagram for the logic of routine DECISION is given in Figure 3. DECISION is called by either BUG.CHK or ACTION from the STAR logic. Both of these routines were significantly altered to allow them to interface with the new decision logic routines. Routine DECISION checks to see if this call is a company or section requesting permission to move by testing the value of the



element of array MVARY referred to by the row and column of the call. If it is not a company or section, DECISION calls the appropriate move routine and schedules a RE.MV.STATE for the calling unit in 60 seconds and then returns from routine DECISION. If it is a company or section request, the logic goes on to test the restriction status of the company or section which is located in the TABLE array. If the company or section is not restricted, the appropriate movement routine is called and a message is created and sent informing the battalion of the unit move. If the unit is restricted from moving, the value of CREQST is checked to see if the unit has previously requested permission to move. If a previous request has occurred, a message is printed noting that fact and then the program returns from routine DECISION. If there has been no previous request for permission to move, the logic checks to see if the unit is on the same phase line as the rest of the units of the battalion. If the unit is farther to the rear than the battalion, a message is printed, and the program returns from routine DECISION. If the unit is on the battalion phase line the program checks to see if the unit has been given permission to move. If the unit has been given permission to move, the appropriate movement routine is called and a message is generated and sent informing the battalion of the unit move. If the unit does not have permission to move, a message is generated and sent to the battalion requesting permission to move. After this, the program exits from routine DECISION.

#### K. ROUTINE GEN.MOVE.DECISION.MSG

The SIMSCRIPT code for routine GEN.MOVE.DECISION.MSG is provided in Appendix N. It is called by all the movement



decision logic routines and it creates the message which is sent from one level to another by the flow of the communications logic. This routine first creates a message and then sets up an array called TEXT. A pointer to the TEXT array where the message's content is stored is saved as an attribute of the MESSAGE. After storing all the relevant information either as attributes of the MESSAGE or in the TEXT array, the program returns from routine GEN.MOVE.DECISION.MSG.

#### L. EVENT COMM0.ATTEMPT

The SIMSCRIPT code for event COMM0.ATTEMPT is provided in Appendix X. The flow diagram for the logic of event COMM0.ATTEMPT is provided in Figure 8. Event COMM0.ATTEMPT takes the messages which are created by the decision process and stochastically models their transmission. This event first checks both the sending and receiving UNIT/TANK to insure that both have radios. If either one does not have a radio, it prints an error message and stops the simulation. It then checks to be sure that both UNITS are on the same FMNET. If they are not, it prints an error message and stops the simulation. If they are, and the net is not busy, it sets the net to busy and calls routine TECH.COMMO which checks to see if communication between the two radios is technically possible. If the net is busy, it files the MESSAGE in the MAILBAG for that SUBSCRIBER and returns from event COMM0.ATTEMPT.

#### M. ROUTINE TECH.COMMO

The SIMSCRIPT code for routine TECH.COMMO is provided in -



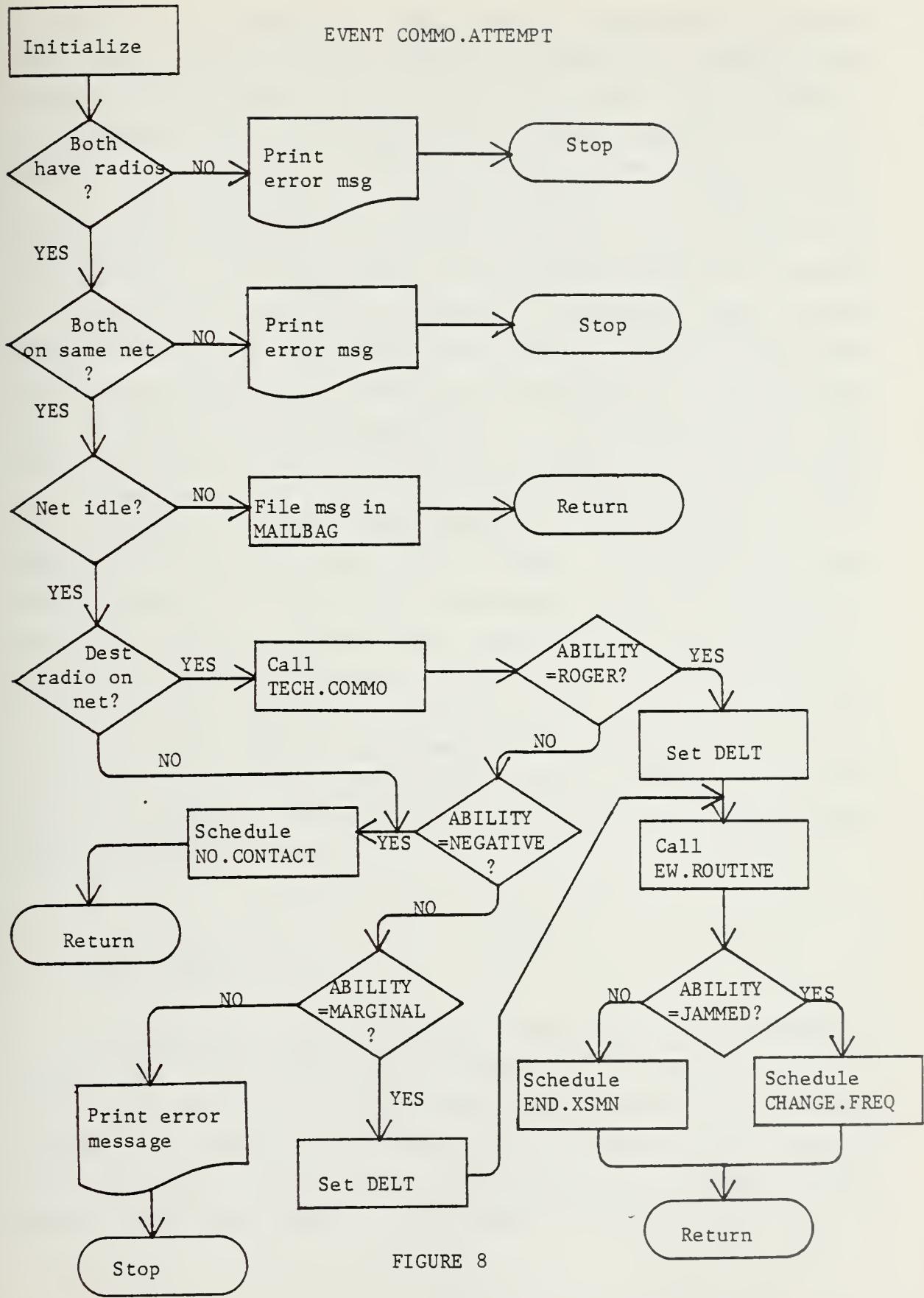


FIGURE 8



Appendix Y. Routine TECH.COMMO determines whether communication between two specific radios is technically possible. It returns one of four values for the ABILITY of the communication. It is either ROGER, meaning good, or MARGINAL, or JAMMED, or NEGATIVE meaning there was no answer by the distant station. If the ABILITY is JAMMED, event CHANGE.FREQ is scheduled and the program returns from event COMMO.ATTEMPT. If the ABILITY is NEGATIVE, event NO.CONTACT is scheduled and the program returns from event COMMO.ATTEMPT. If the ABILITY is either ROGER or MARGINAL, routine EW.ROUTINE is called to determine if the message will be jammed during transmission. If the EW.ROUTINE call indicates that the transmission is jammed, event CHANGE.FREQ is scheduled and the program returns from event COMMO.ATTEMPT. If the transmission is not jammed, event END.XSMN is scheduled and the program returns from event COMMO.ATTEMPT. It currently generates the four possible results by doing a random draw against the four integers 0, 1, 2, and 3. An initial procedure has been written to generate these values for ABILITY by a detailed algorithm based upon the radio characteristics, the terrain, and the locations of the two SUBSCRIBERS. This detailed algorithm will be enhanced and eventually it will replace the random draw currently in use.

#### N. EVENT CHANGE.FREQ

The SIMSCRIPT code for event CHANGE.FREQ is provided in Appendix Z. Event CHANGE.FREQ simulates the delay caused by having to change the net frequency as a result of its being jammed. It sets the attributes of the net and the calling radio to idle and then reschedules the COMMO.ATTEMPT. Because the scheduling of the CHANGE.FREQevent includes the



delay time that would be caused by changing the frequency, the COMMO.ATTEMPT is rescheduled "now." Also, because the probability of jamming is independent and there is no memory of the frequency, the frequency is not actually changed.

#### O. ROUTINE EW.ROUTINE

The SIMSCRIPT code for routine EW.ROUTINE is provided in Appendix AA. Routine EW.ROUTINE simulates the possibility of a transmission being detected and jammed by enemy action. At present, this is a random draw from 0, 1, 2, and 3 with equal probabilities. A draft module to model the direction finding and jamming activities explicitly has been written and it will be enhanced and will eventually replace the current uniform random draw.

#### P. EVENT NO.CONTACT

The SIMSCRIPT code for event NO.CONTACT is provided in Appendix BB. Event NO.CONTACT simulates the occurrence of no response by the distant station. It sets the status of the net and the calling radio to idle and then calls routine SIEZE.NET before filing the current MESSAGE in the MAILBAG of the calling radio.

#### Q. ROUTINE SIEZE.NET

The SIMSCRIPT code for routine SIEZE.NET is provided in Appendix CC. Routine SIEZE.NET does a random draw against all the SUBSCRIBERS on a specific FMNET which have MESSAGES waiting in their MAILBAGS to determine which of them will



get the net which has been left idle as a result of the occurrence of either a NO.CONTACT or END.XSMN event. It then removes that MESSAGE from its MAILBAG and schedules a COMM0.ATTEMPT "now."

#### R. EVENT END.XSMN

The SIMSCRIPT code for event END.XSMN is provided in Appendix DD. The flow diagram for the logic of event END.XSMN is provided in Figure 9. Event END.XSMN simulates the arrival of the MESSAGE at the distant station. It gathers the message identification data into global variables and then destroys the MESSAGE. It then tests the value of the destination code of the message to determine which level of the logic should handle the MESSAGE. This is possible since every MESSAGE has both a type and a destination code and because the text of the message where the data for the decision process is stored is set up as a separate variable length record for which only a pointer is stored in the actual MESSAGE. This means that each type message can have a different length text, and the contents can be generated separately from the actual fixed format portion of the MESSAGE. The attributes of each MESSAGE are passed as arguments through the communications process. A diagram of the pointer links used in the communications logic is provided in Figure 2. Based upon the destination code, one of the message distribution routines, CO.MSG, BN.MSG, BDE.MSG, or DIV.MSG is called. It then releases the text where the MESSAGE data was stored and sets the status of the calling radio, the receiving radio, and the FMNET to idle. It then checks the simulation time using the FORTRAN routine MYTIME to see if the allocated CPU time for the execution of the program is about to run out.



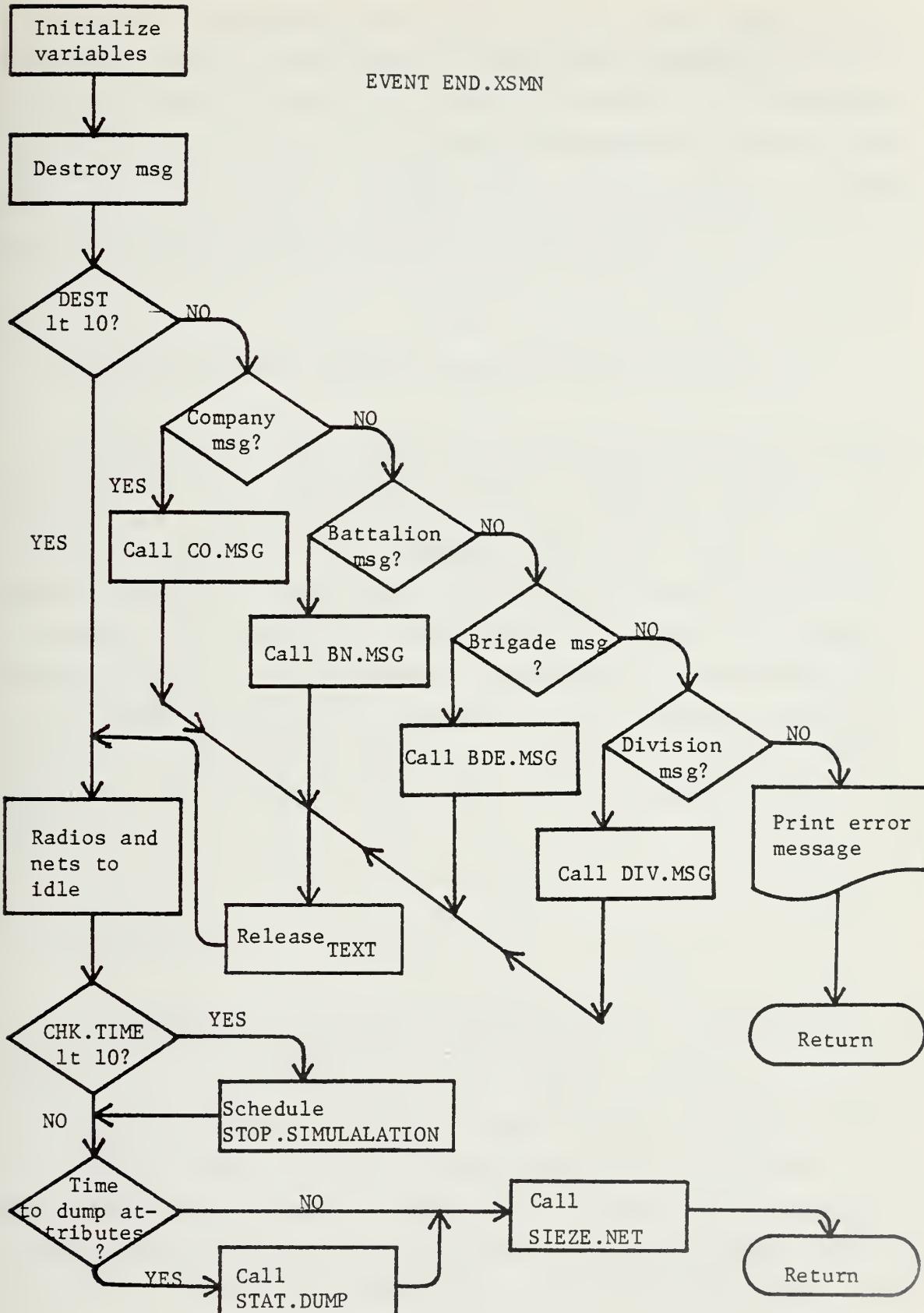


FIGURE 9



If less than 10 seconds of time remain, a STOP.SIMULATION is scheduled "now." Regardless of the time remaining, the time since the last printout of the attributes of the commanders is checked and if it is has been approximately (within 10%) COM.DATA.DUMP seconds since the last print, routine STAT.DUMP is called. Finally, routine SIEZE.NET is called and the program returns from event END.XSMN.

#### S. ROUTINES CO.MSG, BN.MSG, BDE.MSG, and DIV.MSG

The SIMSCRIPT code for all of these routines is provided in Appendix EE. Routines CO.MSG, BN.MSG, BDE.MSG, and DIV.MSG all act as distribution activities which test the value of the type of the MESSAGE and call the appropriate logic routine to handle that MESSAGE. In future versions of the model there will be many different types of messages corresponding to the different functions to be accomplished by the messages such as calling for air support, artillery fires, or logistics support. Routines DEC.CO, DEC.BN, DEC.BDE, and DEC.DIV are all called from these .MSG routines.

#### T. ROUTINE DEC.CO

The SIMSCRIPT code for routine DEC.CO is provided in Appendix I. In addition, a flow diagram of the logic for routine DEC.CO is provided in Figure 4. Routine DEC.CO handles the movement decision logic at the company level. It is called from routine CO.MSG based upon the value of the MSG.TYPE attribute of the MESSAGE. It first stores the text of the MESSAGE in global variables and then tests for the specific type of the MESSAGE. The MESSAGE types are



delineated in the PREAMBLE as global "DEFINE TO MEAN" statements. Based upon the value of the MSG.TYPE, several different actions are taken. If the MESSAGE restricts the company from moving, the company's permission attribute is set to zero, a line noting that the company has been ordered not to move is printed and the program returns. If the MESSAGE gives the company permission to move, the company's permission attribute is set to one and a line noting that the company has been given permission to move is printed before further processing of the MESSAGE is attempted. If the MESSAGE orders the company to move, the company's permission attribute is set to one, a line noting that the company has been ordered to move is printed, and then movement of each TANK in the company is initiated. Routine AIR.COMMO and routine CO.GO are called for each TANK and a RE.MV.STATE is scheduled for the company in 60 seconds. At this point, the value of ORDER is set to two and routine EXIT is called prior to the return from the DEC.CO routine. If the message falls past the above described procedures which provide returns, it then causes the location of the company to be determined. Then it checks to see if the company has been given permission to move. If it has, ORDER is set to one. If not, ORDER is not reset from the value it has on entry to this section of code. Then routine EXIT is called and the program returns.

#### U. ROUTINE DEC.BN

The SIMSCRIPT code for routine DEC.BN is provided in Appendix J. A flow diagram of the logic for routine DEC.BN is provided in Figure 5. This routine handles the movement decision logic at the battalion level. It is called from routine BN.MSG based upon the value of the MSG.TYPE



attribute of the MESSAGE being processed. It stores the text of the MESSAGE in global variables and then tests for the specific type of the MESSAGE. If the MESSAGE is a request to move from a company which has not previously requested permission to move, the value of COWT is added to BNCUR and the request count of the company is set to one before any furthur processing is attempted. It then checks to see if the MESSAGE is a notification of a company or section move. If it is a company movement notification, a line noting that fact is printed and then a MESSAGE is generated and sent to inform the brigade that a company has moved. If it is a section movement notification, a line noting that fact is printed and the program returns. If the MESSAGE is an order to the battalion to move, the battalion's permission attribute is set to one and the BNGO value is added toe the BNCUR value prior to further processing. If the MESSAGE is permission for the battalion to move, the battalion's permission attribute is set to one and the BNCUR value is set equal to the BNLO threshold. If the MESSAGE orders the battalion not to move, the battalion's permission attribute is set to zero and the program returns without any further processing. When a MESSAGE reaches this point, the originator and addressee are interchanged to facilitate subsequent responses. At this point, the permission status of the battalion is checked. If it is a one, the battalion has permission to move and subsequent tests are done to see if the BNCUR value has exceeded the BNLO or the BNGO thresholds. First, the BNGO bound is checked and if it has been exceeded, a MESSAGE is generated and sent to each company in the battalion crdering the companies to move. Also, a line is printed stating that the companies have been ordered to move and a MESSAGE is generated and sent informing the brigade that the companies have all been ordered to move. If the BNLO threshold has



been exceeded, each company is sent a MESSAGE which gives it permission to move and a line is printed noting that fact. Also, a MESSAGE is sent to the brigade with the information that all the companies have been given permission to move. If the BNCUR value doesn't exceed BNLO, the company is sent a MESSAGE ordering it not to move and a line is printed noting this action. If the battalion is restricted from moving and the BNCUR value exceeds the BNLO bound, the battalion sends a MESSAGE to the brigade requesting permission to move. Finally, if the battalion is restricted and the BNCUR value does not exceed the BNLO bound, the battalion sends a MESSAGE to the originating company ordering it not to move. Then the program returns from routine DEC.BN.

#### V. ROUTINE DEC.BDE

The SIMSCRIPT code for routine DEC.BDE is provided in Appendix K. A flow diagram of the logic for routine DEC.BDE is provided in Figure 6. This routine handles the movement decision logic at the brigade level. It is called from routine BDE.MSG based upon the value of the MSG.TYPE of the MESSAGE being processed. The logic of this routine is exactly symmetric to the logic in routine DEC.BN with battalion, brigade, and division substituted for company, battalion, and brigade respectively. The attributes used must be those for the brigade vice those for the battalion.

#### W. ROUTINE DEC.DIV

The SIMSCRIPT code for routine DEC.DIV is provided in Appendix L. A flow diagram for the logic of routine DEC.DIV



is provided in Figure 7. This routine handles the movement decision logic at the division level. It is called from routine DIV.MSG based upon the value of the MSG.TYPE of the MESSAGE being processed. The logic of this routine is the same as that at the battalion and brigade level except that it is somewhat truncated since no messages are sent to a higher command. The attributes used must be those for the division level and brigade and division must be substituted for battalion and brigade.

#### X. EVENT DUMP.MAILBAG

The SIMSCRIPT code for event DUMP.MAILBAG is provided in Appendix FF. Event DUMP.MAILBAG simulates the event of going through all the MAIL.BAGs of all the SUBSCRIBERS and checking to see if the time of the simulation has exceeded the LST.SEND.TIME for any of the MESSAGES in any of the MAILBAGs. Each MESSAGE which is too old is removed from its MAILBAG and routine ABORT.MSG is called for it. Event DUMP.MAILBAG then reschedules itself in 600 seconds.

#### Y. ROUTINE ABORT.MSG

The SIMSCRIPT code for routine ABORT.MSG is provided in Appendix GG. Routine ABORT.MSG simulates the action of sending a MESSAGE by another means if it gets to be a certain age rather than to continue to wait for the transmission. Currently, it simply destroys the MESSAGE, but, it is planned to upgrade this routine so that it will simulate the transmission by courier or some other means explicitly.



## Z. EVENT MSG.GEN

The SIMSCRIPT code for event MSG.GEN is provided in Appendix HH. Event MSG.GEN simulates the background noise messages which monopolize the communications nets and with which the movement decision messages must compete for time on the nets. It generates messages at random on all the simulated nets. It first checks to insure there are at least two SUBSCRIBERS on the net. Then it selects at random one of these SUBSCRIBERS to be the originator of the MESSAGE and another of them to be the destination of the MESSAGE. It then creates the MESSAGE, sets its attributes, increments the NXT.MSG.NO for the originating station, and schedules the COMMO.ATTEMPT to send the MESSAGE. It then selects at random the next FMNET on which a "noise" MESSAGE will be sent and reschedules itself to generate the MESSAGE on that net.



## VII. COMMUNICATIONS MODULE PARAMETERIZATION

There are several features of the communications parameterization process which warrant detailed examination and they are covered in the paragraphs below.

The communications between any two receivers in the same net depends on several variables. One of these variables is the ability of the radios to send a signal of sufficient strength so that it is greater than the squelch threshold of the distant receiver. This is a function of radio line-of-sight, polarity of the antennas, power output of the radios, and many other factors. In the present implementation of communications, this capability is modeled as a uniform random draw against an integer variable between 0 and 3. An ability of 0 means the communication is feasible. An ability of 1 means the communication is marginal and requires two complete transmissions to get the message through. The effect of this is to tie up the net for twice as long as if the ability had been 0. An ability of 2 means the distant station does not answer. An ability of 3 means that successful jamming was encountered and a frequency change is scheduled prior to attempting to send the message again. Once the ability is determined by routine TECH.COMMO, there is another factor which must be considered when the message is actually being transmitted, the ability of the enemy to selectively detect and jam the communications. In the current version of the communications model, this ability is modeled as a uniform distribution with a  $1/4$  probability of jamming the transmission and a  $3/4$  probability of no jamming. In the final version of the communications model the distribution of the jamming will be a user input which can be any



suitably defined distribution function. In addition, it is anticipated that a module will be used to calculate the transmission loss using the STAR LOS (line of sight) routine, the power and transmission characteristics of the radios, and the data on the antennas. This transmission loss will be translated into a 0, 1, 2, or 3 to give a clear, a partial, a blocked or a jammed transmission respectively.

The ABORT.MSG routine is currently used to drop old messages from the lists of those waiting to be sent. In subsequent revisions of this logic, it is anticipated that an alternate method of delivery for these messages will be developed, and they will be sent by this alternate means if they get to be a certain age. Currently, they are simply destroyed if they reach the designated age without being transmitted. The age at which the destruction is invoked is a user input.

The user has two important parameters at his disposal to assist him in debugging any problems in the communications code or the input data. The first of these, COM.PRINT, determines which of the builtin messages are printed out by the program during execution. There are several levels for this parameter. It is tested for being greater than 10, 15, 20, and 25 at various points in the logic. The larger the value of this parameter used, the more detailed will be the printout of data during the execution. The second parameter, COM.DATA.DUMP, determines the amount of simulated time which is allowed to pass before a dump of the attributes of the company, battalion, brigade, and division commanders is executed. This attribute dump enables the user to check to be sure that the correct weights are being used for the units and that the request and permission attributes of the commanders are being properly set by the program. Both of these parameters are read in from cards at the beginning of the STAR BIG.MAIN routine.



## VIII. COMMUNICATIONS MODULE INPUT REQUIREMENTS

The input requirements for the communications module fall into several categories based upon which module reads them in and uses them. The following descriptions are given in the same order as they occur when executing the model and are broken down into sections by routine. All read statements use the free form read with at least one blank separating variables.

### BIG.MAIN

#### A. Read COM.PRINT COM.PRINT

This integer variable determines the level of detail which is printed out during execution of the model. When set to zero, all of the print statements are turned off. The major break points in the printout decision logic are set at 10, 15, 20, and 25. Any COM.PRINT value larger than these levels causes all those levels below it to be turned on.

#### B. Read COM.DATA.DUMP COM.DATA.DUMP

This real variable determines how often the values of the attributes of the unit commanders are printed out. For example, if read in as 600.0, approximately every 600 seconds of simulation time the attributes will be printed.

### BUILDNETS

#### A. Read NONETS, NORELAY NONETS

This integer variable specifies the total number of FM nets which are to be simulated.

NORELAY

This integer variable specifies the total number of relays to be simulated.

#### B. Read NFMN, NRSUB, NALTFR NFMN

This integer variable specifies the number designator of the *i*th radio net.

NRSUB

This integer variable specifies the number of alternate frequencies assigned to the *i*th radio net.

NALTFR

This integer variable specifies the number of alternate



frequencies assigned to the ith radio net.

C. Read NFCK and NPRI

NFCK

This integer variable is a check to insure that the correct frequency is assigned to each net. It must be the same as NFMN from B. above to avoid an error.

NPRI

This integer variable is the primary frequency assigned to the ith net.

D. Read TKHOLD

TKHOLD

This integer variable is the number designator of the UNIT/TANK which belongs to the SUBSCRIBER created in this loop. This SUBSCRIBER will be a node in the net designated by NTCK and NFMN. All the SUBSCRIBER number designators are read in and the SUBSCRIBERS filed in the net before going on to read statement E.

E. Read NETNO

NETNO

This integer variable must equal NFMN to avoid an error.

F. Read TKALT

TKALT

This integer variable has the alternate frequencies of net NFMN read into it one at a time. All these alternate frequencies are read and stored in the ragged array FM.PUSHES in a loop prior to going to the next read statement B.

## COMMGEAR

A. Read NRADIO.TANKS

NRADIO.TANKS

This integer variable specifies the number of tanks in the simulation which will have radio equipment.

B. Read TKNO, NORAD

TKNO

This integer variable specifies the number designator of the TANK/UNIT which will have communications equipment.

NORAD

This integer variable specifies the number of sets of radio equipment (RT.CONFIG's) the TANK/UNIT will have.

C. Read RAD.TYPE(RT.CONFIG) and MODE.OPERATION(RT.CONFIG)

RAD.TYPE(RT.CONFIG)

This integer attribute of RT.CONFIG specifies the type of radio in use. At present these are dummy types and have no real significance.

MODE.OPERATION(RT.CONFIG)

This integer attribute of RT.CONFIG specifies the mode of operation of the radio. At present these are dummy modes and have no real significance.

D. Read ANT.TYPE(RT.CONFIG)

ANT.TYPE(RT.CONFIG)

This integer attribute of RT.CONFIG specifies the type of antenna being used for this radio set. If this number is less than zero, the antenna is directional,



and since there are no provisions for this type in the rest of the program at this time, this causes the program to print a message concerning directional antennas and stop.

E. Read REM.ANT(RT.CONFIG)  
REM.ANT(RT.CONFIG)

This integer attribute of RT.CONFIG specifies whether or not the antenna is remoted from the radio. If the value is not zero, the antenna is remoted, and since there is no provision for this in the rest of the program, it causes the program to print a message concerning remoted antennas and stop.

## INITIALIZE

A. Read PAR00, PAR01, ... PARxx  
PARxx

These are integer parameters used for generating random numbers using the SIMSCRIPT random number calls. The value of xx should not exceed 49, and each should be different.

B. Read PAR50, PAR51, ... PARyy  
PARyy

These are real parameters which are used for random number generation. The yy value should be larger than 49, and each yy must be unique.

C. Read CO.LO.BD and CO.UP.BD  
CO.LO.BD

This integer variable is the lower bound of the destination code for messages being sent to the company level.

CO.UP.BD  
This integer variable is the upper bound of the destination code for messages being sent to the company level.

D. Read BN.LO.BD and BN.UP.BD  
BN.LO.BD

This integer variable is the lower bound of the destination code for messages being sent to the battalion level.

BN.UP.BD  
This integer variable is the upper bound of the destination code for messages being sent to the battalion level.

E. Read BDE.LO.BD and BDE.UP.BD  
BDE.LO.BD

This integer variable is the lower bound of the destination code for messages being sent to the brigade level.

BDE.UP.BD  
This integer variable is the upper bound of the destination code for messages being sent to the brigade level.

F. Read DIV.LO.BD and DIV.UP.BD  
DIV.LO.BD

This integer variable is the lower bound of the destination code for messages being sent to the division level.

DIV.UP.BD



This integer variable is the upper bound of the destination code for messages being sent to the division level.



## IX. FUTURE MODEL ENHANCEMENTS

The model of communications described in this document represents a good foundation for developing those additional communications-related modules required to enhance the realism of the STAR Model. The structure of this model is sufficiently flexible to handle most of the communications requirements envisioned for inclusion in STAR.

Obviously, there is still much to be done. A few of the more important areas for future work are identified in this chapter.

Enhancement of the red interception and jamming capability module initially developed by Cpt. Haislip is high on the list of significant actions which will have a marked effect on the realism of the current model since it incorporates a call to a jamming routine for which the user supplies parameters.

In addition to the red electronic warfare capabilities, the blue force EW assets and doctrine must be evaluated and modeled to provide a balanced picture of the impact of EW on the model.

The play of relays for the FM nets, radio wire integration, radio teletype HF nets, and directional antennas should be considered for inclusion in the model. It may be that some of these do not provide sufficient gains in realism to justify their being included, but they should be studied.

Since the STAR model is headed for the goal of modeling a blue brigade under attack by a red division, the multichannel radio nets within the brigade boundary and those going back to the corps level need to be studied and considered for inclusion in the model. These nets represent



a large proportion of the communications assets in the brigade area and should be included in the model in some manner. Additionally, as more and more of the logistics and support functions are included in the model, the need to model communications to entities other than the unit commanders will increase. This means modeling at least part of the communications network down to the individual instrument or line where these support people are located.

Finally, the whole area of Command, Control, Communications, and Intelligence needs to be integrated in some way with the tactical decision logic in order to take advantage of the information which can be gained from the intelligence network.



## APPENDIX A

### STAR ROUTINES, EVENTS, AND ARRAYS

The following routines, events, and arrays are key elements in the existing STAR movement decision logic.

#### ROUTINES

##### Action

This routine calls the movement decision routine and takes action based on the value of the order returned to initiate movement of units in conjunction with the leave logic.

##### Bn.go

This routine initiates movement of a blue battalion from its current position.

##### Bug.chk

This routine checks to see if a unit should move as a result of being too close to an enemy unit and calls the movement decision routine if required.

##### Co.go

This routine initiates movement of a blue company from its current position.

##### Decision

This routine is called by routine ACTION or routine BUGCHK. It updates the values of CREQST, BREQST, BNCUR, and BDECUR and checks the attributes of a unit and the values stored in the arrays TABLE, BNCORD, and COCORD to determine if the unit is allowed to perform a desired movement. It causes units to move if the BNCUR or BDECUR values exceed the LO or GO bounds for the battalion or the brigade.

#### EVENTS

##### Phaz.chk

This event determines whether the current phase line is still occupied. If it is still occupied, the event reschedules itself. If it isn't occupied, the event removes the battalion commander from the BRIGADE (set) and calls routine COORD.SET.

##### Re.mv.state

This event checks to see if units have arrived at their new locations.



## ARRAYS

### Table

A diagram of the organization of this array is provided in Figure 10. This three dimensional array has a plane for each blue maneuver unit and seven rows for each plane. The first six rows correspond to monitored weapons systems for the unit and the seventh corresponds to the unit as a whole. Each weapon system row has 4 plus  $3 * \text{NPRI}$  columns where NPRI is the number of attrition level actions planned for that system. Column 1 contains the system code in the first 6 rows and the boundary range in the seventh row. If no enemy units are within this range, the unit will not move off its position unless ordered to do so. Column 2 contains the weapon code in the first six rows and the range within which all enemy units are used for computing the force ratio in the seventh row. Column 3 contains the closing range within which an enemy firing at a system causes it to request permission to move off its position. Column 4 contains the code for whether a system or unit is restricted from moving or is free to move at will. The remaining columns are in groups of 3 with the first being an attrition level, the second being the force ratio, and the third being the code for the action to be taken if that attrition level and force ratio are exceeded.

### Bnord

A diagram of the organization of this array is provided in Figure 11. This three dimensional array has a plane for each blue battalion, a row for each phase line, and five columns. The first column contains BMSN, the code for whether or not the battalion has permission to move off this coordination line. The second column contains BNCUR, the sum of the weights of the companies which have requested permission to move off their current coordination lines. The third column contains BNWT, the tactical weight of the battalion on this coordination line. The fourth column contains BNLO, a lower bound which if exceeded by BNCUR causes the battalion to give all the companies permission to move. The fifth column contains BNGO, a lower bound which if exceeded by BNCUR causes the Battalion to order all companies to move.

### Cocord

A diagram of this array is provided in Figure 12. This three dimensional array has a plane for each blue company unit, a row for each coordination line, and two columns. The first column contains CMSN, the code for whether or not the company has permission to move off its current coordination line. The second column contains COWT, the tactical weight of the company on this coordination line.



TABLE DIAGRAM

The diagram illustrates a 3D perspective view of a cube representing a table for maneuver unit performance. The cube has three dimensions: System Type (vertical), Weapon Type (depth), and Range (width).

**Labels:**

- Action:** Points to the first column of the top face.
- Force Ratio:** Points to the second column of the top face.
- % Attrition:** Points to the third column of the top face.
- Action:** Points to the first column of the bottom face.
- Force Ratio:** Points to the second column of the bottom face.
- % Attrition:** Points to the third column of the bottom face.
- Restriction Status:** Points to the fourth column of the bottom face.
- Range:** Points to the fifth column of the bottom face.
- System Type:** Points to the sixth column of the bottom face.
- Unit #n**: Points to the vertical axis of the cube.
- Unit #2**: Points to the second column of the top face.
- Unit #1**: Points to the first column of the top face.

**Data on the Top Face:**

Action	Force Ratio	% Attrition									
1	1	800	1	75	3	72	75	2	71	32	41
1	2	800	1	50	3	71	50	2	42	99	50
1	3	1000	0	50	3	32	42	2	62	99	42
2	4	1200	0	99	.				71	50	2
2	0	0	0	50	2				42	99	42
3	6	0	0	50	3				62	99	42
5	1	1200	1	75	3				71	50	2
3000	2500	800	1								

**Data on the Bottom Face:**

Range	System Type	Weapon Type	Restriction Status	Action	Force Ratio	% Attrition
1	1	800	1	75	3	72
2	2	800	1	50	3	71
3	3	1000	0	99	.	32
4	4	1200	0	50	2	42
5	0	0	0	50	3	41
6	6	0	0	50	2	42
7	1	1200	1	75	3	42
8	1	2500	1			

A plane for each blue maneuver unit

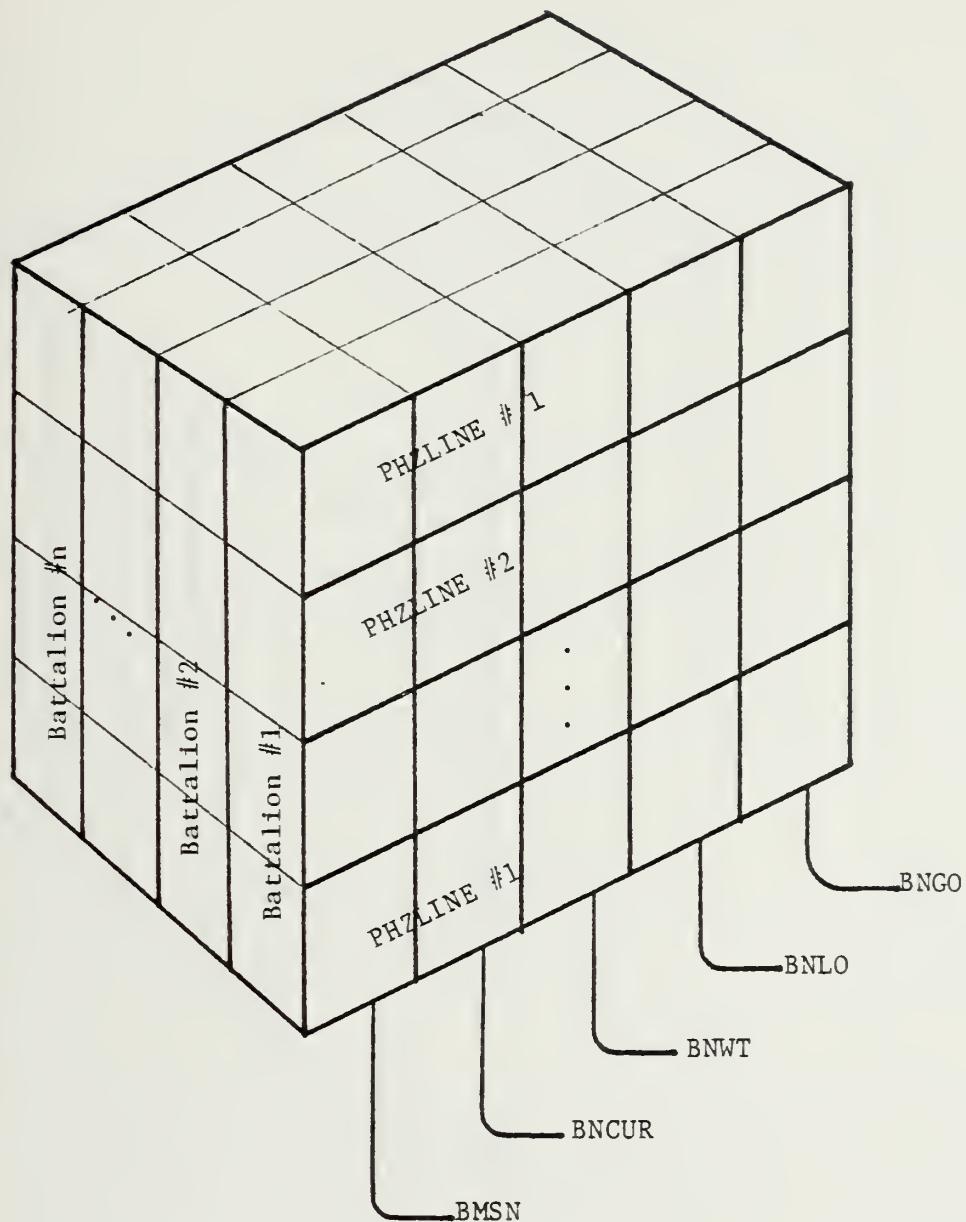
A column in each plane for each of the above weapon system attributes

A row in each plane for each of up to seven weapon systems

FIGURE 10



BNCORD DIAGRAM



A plane for each battalion

A row in each plane for each phase line

A column in each plane for each of the five attributes

FIGURE 11



COCORD DIAGRAM

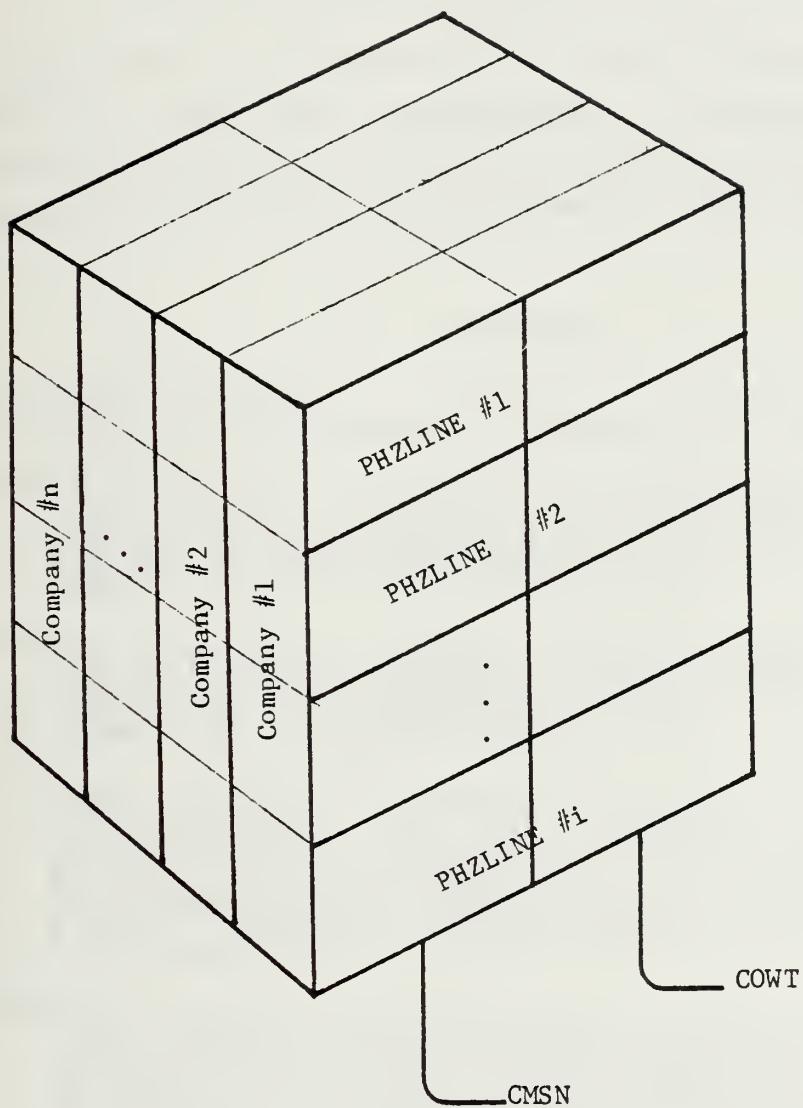


FIGURE 12



## APPENDIX B

### COMMUNICATIONS ROUTINES AND EVENTS

The communications model uses the following routines and events to model the transmission of messages between the decision levels. The decision process is based upon the information contained in the messages and their texts.

#### ROUTINES

**Abort.msg(tank, msg)**  
This routine destroys a message if it gets too old.

**Bde.msg**  
This routine is called by event END.XSMN whenever a brigade-level message is encountered. This routine in turn calls the appropriate logic routine to handle the message based on the message type.

**Bn.msg**  
This routine is called by event END.XSMN whenever a battalion-level message is encountered. This routine in turn calls the appropriate logic routine to handle the message based on the message type.

**Bnettbl**  
This routine is called by routine COM.MAIN during initialization of the program. It builds two two-dimensional tables or ragged arrays. The first, NETTBL, stores the subscripts of the subscribers of the Ith radio net in row I of the array. The second, RPOINT, stores in its Ith row the address pointers of the subscribers in the Ith radio net.

**Buildnets**  
This routine is called by routine COM.MAIN during initialization of the program. It reads in the number of radio nets, the number of relays, the subscribers in each net, and the alternate frequencies for each net. It creates the FMNETS and files them in the ETHER and also creates the SUBSCRIBER's and files them in the CEOI.LIST of the FMNET to which they belong.

**Checknets**  
This routine is called by routine COM.MAIN during initialization of the program. It cycles through each FMNET and prints the number of the net, the primary frequency of the net, the number of alternate frequencies, the alternate frequencies, and the number of SUBSCRIBER's in the net. This is primarily a data debugging facility.

**Co.msg**



This routine is called by event END.XSMN whenever a company-level message is encountered. This routine in turn calls the appropriate logic routine to handle the message based on the message type.

#### Com.main

This routine calls the routines which initialize the communications portion of the program. It then schedules the DUMP.MAILBAG and MSG.GEN(FMNET) events and returns control to the SIMSCRIPT II.5 scheduler.

#### Commcheck

This routine is called by routine COM.MAIN during initialization of the program. It cycles through every TANK in BLUE.ALIVE and prints out the communications gear which has been assigned to each TANK, the location of each TANK, the mode of operation of each radio, and the type of antenna used by each radio. This is primarily a data debugging facility.

#### Commgear

This routine is called by routine COM.MAIN during initialization of the program. It reads the number of each TANK which is to have communications capability and the number of radios each TANK is to have. It then creates a MYSET for that TANK which is a set containing pointers to all the RT.CONFIG's (radios) owned by that TANK. It creates the required number of RT.CONFIG's for that TANK and files them in RAD.LIST of MYSET. It reads the radio type, the mode of operation of the radio, and the antenna type used by the radio and sets these attributes of each of the RT.CONFIG's created. After creating all of the radios, the routine sets the next message number of each tank to n001, where n is the number of the TANK. It then places the equipment in the appropriate net, and prints out a matrix of the TANK subscript and the address pointers for the TANK, the MYSET, the RT.CONFIG, and the SUBSCRIBER for each TANK in each FM net in the ETHER. This listing is primarily a debugging facility.

#### Dec.bde

This routine handles the movement decision at the brigade level. It checks to see if a battalion should move or be given permission to move and if so, it causes the appropriate message to be generated and schedules the COMMO.ATTEMPT to the battalion.

#### Dec.bn

This routine handles the movement decision at the battalion level. It causes messages to be generated and schedules COMMO.ATTEMPT's to both the company level and the brigade level.

#### Dec.co

This routine initiates movement of the company from its current position by calling the CO.GO routine if authorized by the text of the message.

#### Dec.div

This routine handles the movement decision logic at the division level. It causes messages to be generated and schedules COMMO.ATTEMPT's to the brigade level.

#### Decision(row,col,t)

This routine does the preliminary checks to determine if a company needs permission to move and if so, it



causes a message to be generated and schedules a COMMO.ATTEMPT to the battalion. If permission is not required, it causes the unit to begin movement.

#### Div.msg

This routine is called by the END.XSMN event whenever a division-level message is encountered. This routine then calls the appropriate logic routine to handle the message based on the message type.

#### Ew.routine

This routine is called by the COMMO.ATTEMPT event whenever a message is being placed on the net to determine if the message will be jammed. If the message is jammed, event CHANGE.FREQ is scheduled and executed prior to allowing another attempt to send the message.

#### Exit(order,t,crit,p.z.time,flag)

This routine is used as an escape from the decision process if permission is not required for a company to move from its current position.

#### Gen.move.decision.msg(pharow,order,t,crit,p.z.time) yielding msg

This routine produces the movement request and order messages, assigns values to the message text variables, and passes a pointer to the message back to the calling routine.

#### Initialize

This routine is called by routine COM.MAIN during the initialization of the communications portion of the program. This routine reads the parameters for the distributions used in scheduling the communications events. It also calls routines SSTAT.DUMP and STAT.DUMP.

#### Sieze.net(callnet)

Upon termination of a message on any of the FMNET's, this routine checks that FMNET to see if there are any messages which have been saved and need to be transmitted over it. If so, it selects one at random and schedules the COMMO.ATTEMPT for that message.

#### Sstat.dump.

This routine is called by routine INITIALIZE during the initialization of the program. This routine prints out the values of the attributes of the brigade, battalion, and company commanders which are used for movement coordination.

#### Stat.dump

This routine is called by routine INITIALIZE during the initialization of the program and by routine SNAP.R if the program should fail. This routine lists the attributes of each company commander, battalion commander, brigade commander, and division commander.

#### Tech.commo(callrt,recrt) yielding ability

This routine is used to determine how good communication is between the origin and destination stations attempting to pass and receive each message. A message can be one of four types: 1) normal transmission, 2) marginal transmission (It takes twice as long as a normal transmission.), 3) no response (The distant station does not answer.), and 4) jammed transmission (The sender experiences jamming as soon as



he enters the net and immediately schedules a CHANGE.FREQ as a result.).

Xy.antenna(rt.config) yielding xt, yt, and zt  
This routine determines the x, y, and z coordinate locations of remote antennas. It is called by routine COMMGEAR during the initialization phase of the program.

## EVENTS

Change.freq(callrt,callnet,msg)

This event is called when the attempted transmission is jammed. It "changes" the transmitting frequency of the net. In fact, the frequency isn't changed since there is no memory property for the jamming routine and each test to see if a MESSAGE is jammed is an independent trial. The frequency could be changed by a quite simple algorithm, but since this isn't required, another COMMO.ATTEMPT is scheduled after a suitable wait representing the time required to change the frequency.

Commo.attempt(orig,msg)

This event simulates the process of acquiring communications with a distant station. It checks to insure that both UNIT's/TANK's have radios, that they are both on the same net, that communication is technically feasible, that the net is not busy, and that the distant station responds to the call. It files the MESSAGE in a MAILBAG if it can't be transmitted and schedules another COMMO.ATTEMPT at a later time. It also sets the busy-state of the FMNET to busy.

Dump.mailbag

This event checks every station in every net for messages waiting to be transmitted which have a LST.SEND.TIME attribute less than simulation time. When it finds a MESSAGE which satisfies this condition, it removes the MESSAGE from the MAILBAG of those waiting for transmission and calls ABORT.MSG which destroys the MESSAGE.

End.xsmn(callrt,recrt,callnet,msg)

This event is scheduled when the COMMO.ATTEMPT for each MESSAGE occurs. It is the current simulated time plus the length of the MESSAGE. When this event occurs, the MESSAGE content is examined and based upon the destination code, one of the DEC. routines is called to handle the decision process. This event also sets the busy-state of the FMNET to idle.

Msg.gen(fmnet)

This event generates "background noise" messages on the nets to simulate the other traffic with which the movement decision messages have to compete for time on the net.

No.contact(callrt,callnet,msg)

This event simulates a lack of response by the distant station by setting the busy-state of the net to idle so that other messages can be passed. It also files the MESSAGE in a MAILBAG for future transmission.



APPENDIX C  
COMMUNICATIONS GLOBAL VARIABLES

INTEGER VARIABLES

**Addr.store** This is a temporary storage location for the address to which a MESSAGE will be sent. It is used to pass the address to the message generation routine rather than increase the number of parameters explicitly passed to the routine.

**Adrsee** This is a temporary location used in the decision logic to store the addressee of an incoming MESSAGE. It is loaded by the END.XSMN event just prior to the destruction of the MESSAGE.

**Bbde** This variable is one that was added to those used in the basic STAR model. It contains the value read in for the number of blue brigades to be played.

**Bde.lo.bd** This is an input parameter set by the user as the lower bound of the DEST.CODE of a MESSAGE that will be sent to the brigade.

**Bde.up.bd** This is an input parameter set by the user as the upper bound of the DEST.CODE of a MESSAGE which will be sent to the brigade.

**Bn.lo.bd** Same as BDE.LO.BD but for the battalion level.

**Bn.up.bd** Same as BDE.UP.BD but for the battalion level.

**Co.lo.bd** Same as BDE.LO.BD but for the company level.

**Co.up.bd** Same as BDE.UP.BD but for the company level.

**Com.print** This input parameter determines what information is printed during an execution of the model. The larger COM.PRINT is, the more detailed the printout becomes. When set to zero, the execution printout is minimized. Current break points for increasing the printed output are set at 10, 15, 20, and 25. Any value larger than any of these will cause that level of print to be activated.

**Dest** This is a temporary location used to store and test the destination level of a MESSAGE. The value is set in event END.XSMN just prior to destruction of the



MESSAGE. Subsequent to the MESSAGE's destruction, DEST is tested to determine which command level must take action on the MESSAGE.

Div.lo.bd

Same as BDE.LO.BD but for the division level.

Div.up.bd

Same as BDE.UP.BD but for the division level.

Dvn

This variable was added to those in use by the STAR model. The number of divisions to be simulated is read into it during initialization of the model.

Flag

This is a temporary storage location used by the print routines. It contains an integer value which uniquely identifies the exit which was taken from the movement decision logic. It is used primarily as an aid to debugging.

Msg

This is the core location of the current MESSAGE. It is a pointer which uniquely identifies the MESSAGE.

Msgno

This is a temporary storage location for the number which is used to distinguish one message between two entities from another. This variable is set by the END.XSMN event just prior to destruction of the MESSAGE.

Orig

This is a temporary storage location for the pointer to the originating UNIT of a MESSAGE. It is set in routine END.XSMN just prior to destruction of the MESSAGE.

Parxx

These are input parameters which are used in the calls to various distributions for the purpose of generating random numbers which are used to schedule events. In general, if xx is between 00 and 49, this should be an integer number.

Text.pointer

This variable is a temporary location used to store the pointer to the TEXT which is associated with a particular MESSAGE. It is set in routine END.XSMN just prior to destruction of the MESSAGE.

Type

This is a temporary location used to store the value of the message type. The value is set in routine END.XSMN just prior to destruction of the MESSAGE, and it is tested in the BN.MSG, BDE.MSG, CO.MSG, and DIV.MSG routines after destruction of the MESSAGE to determine which of the logic routines to call to handle the MESSAGE.

## REAL VARIABLES

Chk.time

This is a variable used to store the value returned



from the Fortran function MYTIME in event END.XSMN. The MYTIME routine returns the number of ten thousandths of a second left before the program uses up its allotted CPU time. This number is divided by 10000 and truncated before being stored in CHK.TIME. If the value stored in CHK.TIME is less than 10, a STOP.SIMULATION is scheduled immediately so that the data up to that point will be printed out and not lost as it would be if the job crashed for exceeding its allocated time.

**Com.data.dump**

This is an input parameter which determines how often the STAT.DUMP routine will be executed to print out the attributes of each CO.COMMANDER, BN.COMMANDER, BDE.COMMANDER, and DIV.COMMANDER. For example, if COM.DATA.DUMP is read in as 600.0 by the user, approximately every 600 seconds of simulation time the attributes will be printed out.

**Com.time**

This variable is used to store the value of TIME.V each time the routine STAT.DUMP is called. It is used to insure that STAT.DUMP is only called once every COM.DATA.DUMP seconds of simulation time.

**Paryy**

These are input variables used as parameters for the distributions provided by the Simscript language for scheduling events and generating random numbers. If yy is a value from 50 to 99, the parameter should be a real variable.

**Rni**

This variable is used in routine SIEZE.NET to select at random the MESSAGE which will get the net from all those stored in MAILBAGs which are assigned to the net.



APPENDIX D  
COMMUNICATIONS PERMANENT ENTITIES

BDE.COMMANDER

**Bdecur** This attribute is the sum of the weights of all the battalions in the BRIGADE which have requested permission to move from the current coordination line. It is reset to zero each time the brigade first occupies a coordination line.

**Bdego** This attribute is a user input upper bound. If BDECUR equals or exceeds this value, all the battalions in the BRIGADE are ordered to move back to their next coordination line position. A distinct BDEGO value is linked to each coordination line which the brigade can occupy.

**Bde1o** This attribute is a user input lower bound. If BDECUR equals or exceeds this value, all the battalions in the BRIGADE are given permission to move at will back to their next coordination line position. A distinct BDELO value is linked to each coordination line which the brigade can occupy.

**Bdemsn** This user input attribute is stored in the three-dimensional array, BDECORD. It indicates whether or not the brigade has permission to move from a given coordination line. A 1 means that the brigade has permission to move and a 0 means that the brigade does not have permission to move.

**Bdreqst** This attribute indicates whether or not a brigade has requested permission to move from its current coordination line position. A 1 means that the brigade has requested permission and a 0 means that the brigade has not requested permission to move.

**Bdewt** This user input attribute indicates the tactical weight or relative importance of the brigade's position on a particular coordination line. There is a distinct BDEWF value for each of the coordination lines which the brigade can occupy.

**Brde** This attribute stores the number designator of the brigade.

**No.bde.unit** This attribute stores the number designator of the UNIT/TANK which is associated with the brigade commander.



## BN.COMMANDER

Batt This attribute stores the number designator of the battalion.

Bmsn This user input attribute indicates whether or not the battalion has permission to withdraw from a particular Coordination line. This attribute is stored in the three-dimensional array, BNCORD. A 1 means that the battalion has permission to move while a 0 means that the battalion does not have permission to move.

Bncur This attribute stores the sum of the weights of the companies which have requested permission to move from their current coordination line position. It is reset to zero each time the battalion first occupies a coordination line.

Bngo This user input attribute is an upper bound. If BNCUR equals or exceeds this value, all the companies in the BATTALION will be ordered to move back to their next coordination line position. A distinct BNGO value is linked to each coordination line which the battalion can occupy.

Bnlo This user input attribute is a lower bound. If BNCUR equals or exceeds this value, each company in the BATTALION is given permission to move at will back to its next coordination line position. A distinct BNLO value is linked to each coordination line which the battalion can occupy.

Bnwt This user input attribute indicates the tactical weight or relative importance of the battalion's position on a given coordination line. There is a distinct BNWT value for each coordination line which the battalion can occupy.

Breqst This attribute indicates whether or not the battalion has requested permission to move from its current coordination line position. A 1 means that the battalion has requested permission to move while a 0 means that the battalion has not requested permission to move.

No.bn.unit This attribute stores the number designator of the UNIT/TANK which is associated with the battalion commander.

## CO.COMMANDER



Cmsn This user input attribute stored in the three-dimensional array, COCORD, indicates whether a company has permission to move from a given coordination line position. A 1 means the company has permission to move while a 0 means that the company does not have permission to move.

Copy This attribute stores the number designator of the UNIT/TANK which is associated with the company commander.

Cowt This user input attribute indicates the tactical weight or relative importance of the company's position on a given coordination line. There is a distinct COWT value for each coordination line which the company can occupy.

Creqst This attribute indicates whether the company has requested permission to move from its current coordination line position. A 1 means that the company has requested permission to move while a 0 means that the company has not requested permission to move.

#### DIV.COMMANDER

Ddiv This attribute stores the number designator of the division.

Divcur This attribute stores the sum of the weights of the brigades which have requested permission to move from their current coordination line position. It is reset to zero each time the division first occupies a coordination line.

Divgo This attribute is a user input upper bound. If DIVCUR equals or exceeds this value, all the brigades in the DIVISION will be ordered to move back to their next coordination line position. A distinct DIVGO value is associated with each coordination line which the division can occupy.

Divlo This user input attribute is a lower bound. If DIVCUR equals or exceeds this value, each brigade in the DIVISION is given permission to move at will back to its next coordination line position. A distinct DIVLO value is associated with each coordination line which the division can occupy.

Divreqst This attribute indicates whether the division has requested permission to move from its current coordination line position. A 1 means that it has requested permission to move and a 0 means that it has not requested permission to move.



**Divwt** This attribute is a user input which indicates the tactical weight or relative importance of the division's position on a given coordination line. There is a distinct DIVWT for each coordination line which the division can occupy.

**No.div.unit** This attribute stores the number designator of the UNIT/TANK which is associated with the division commander.

#### B.MVR.CDR

**Prty** This attribute is used to determine the correct plane of the TABLE array which is to be used for a given UNIT on a given coordination line.

There are many other attributes of the B.MVR.CDRs, but this is the only one which has any importance in the communications model. The rest of the attributes are used for combat analysis of casualties and related statistics.



## APPENDIX E

### COMMUNICATIONS TEMPORARY ENTITIES

#### FMNET

**Fm.idle**  
This attribute indicates whether or not the net is in use, with a code of 1, or idle, with a code of 0.

**Fmalt.freq**  
This user input attribute contains the alternate frequency of the FMNET.

**Fmpri.freq**  
This attribute contains the user input primary frequency of the FMNET.

**Ncs.point**  
This attribute contains a pointer to the TANK/UNIT of the net control station for the FMNET.

**Num.net**  
This attribute indicates the numeric designator of this FMNET.

#### MESSAGE

**Addressee**  
This attribute stores the pointer to the UNIT/TANK which is associated with the commander who is the addressee of the message.

**Dest.code**  
This attribute stores a code value which indicates the level to which the message is being sent. For example, if the value of this variable lies between the user input values for CO.LO.BD and CO.UP.BD, the message will be sent to the company level.

**Length.msg**  
This attribute stores the length of the message in simulated seconds. It is used during event COMM0.ATTEMPT to determine when to schedule the END.XSMN event.

**Lst.send.time**  
This attribute stores the simulated time beyond which the message will not be transmitted. It is used in the DUMP.MAILBAG event to determine if the message should be aborted.

**Msg.no**  
This attribute stores the number of the message in the form nxxx where n is the number of the UNIT/TANK sending the message and xxx is a sequential integer which starts at one and increases one for each message sent by this UNIT/TANK.



**Msg.text**  
This attribute stores the pointer to the text which is created to go with this message.

**Msg.type**  
This attribute indicates the type of the message. For example, a type 30 message is a request for permission to move from the battalion to the brigade.

#### MYSET

**Nxt.msg.no**  
This attribute stores the number designator of the next message which this particular TANK/UNIT will send.

#### RT.CONFIG

**Ant.type**  
This attribute contains a code which describes the type of antenna which is used by the radio configuration. If the value is less than zero, the antenna is directional. If the value is greater than zero, the antenna is omnidirectional.

**Ant.x.loc**  
This attribute contains the x coordinate of the location of the antenna which may not be the same as the location of the TANK/UNIT which uses that antenna because remote antennas are allowed.

**Ant.y.loc**  
This attribute contains the y coordinate of the location of the antenna.

**In.use**  
This attribute contains a code which is set to 1 to indicate that the radio set is in use and is set to 0 when the set is not being used.

**Mode.operation**  
This attribute contains a code which designates the mode of operation of the radio.

**Own.subscriber**  
This attribute contains a pointer to the subscriber which owns the TANK/UNIT to which this RT.CONFIG belongs.

**Rad.type**  
This attribute contains a code which designates the type of radio.

**Rem.ant**  
This attribute contains a code which tells if the antenna for this RT.CONFIG is remoted or not. Any value other than zero indicates a remoted antenna.

#### SUBSCRIBER

**Own.net**



This attribute contains a pointer to the FMNET which is used.

**Own.tank**

This attribute contains a pointer to the UNIT/TANK associated with this subscriber.

**Ifc.waiting**

This attribute contains a code of 1 if there is at least one message filed in this SUBSCRIBER's mailbag waiting to be sent. Otherwise, the value of this attribute is 0.



## APPENDIX F

### COMMUNICATIONS ARRAYS

#### BDECORD

The BDECORD array stores five different attributes of the brigades. It is a three dimensional array which has a plane for each blue brigade. Within each plane there is a row for each phase line which the brigade can occupy and within each row there are five columns. One column for each of the five attributes which will be stored. The first column contains BDEMSN, the code for whether or not the brigade has permission to move off this coordination line. The second column contains BDECUR, which is the sum of the weights of the battalions which have requested permission to move from their coordination line positions. The third column contains BDEWT, which is the user supplied estimate of the tactical importance of the brigade on this coordination line. The fourth column contains BDELO, the lower bound which if exceeded by BDECUR causes the brigade to give all its battalions permission to move from their current coordination line positions. The fifth column contains BDEGO, the upper bound which if exceeded by BDECUR causes the brigade to order all its battalions to move from their current coordination line positions. This array is used to reset the attributes of the brigade commanders after they first occupy a coordination line position. The array itself, with the exception of BDEMSN and BDECUR values is loaded during the initialization of the program using input data supplied by the user. A diagram of the way this array is organized is provided in Figure 13.

#### BNCORD

This array is the same one that is used by the production STAR model. It is explained in Appendix A. A diagram of the organization of the array is provided in Figure 11.

#### BNETTBL

This two dimensional array is used by the communications routines as a table lookup which contains the subscript number of the UNITS/TANKS which belong to the FMNETs. The subscripts of the TANKS which belong to the  $i$ th FMNET are elements of the  $i$ th row of the array. Because the FMNETs are not restricted to all having the same number of SUBSCRIBERS, this array is necessarily ragged. An example of this array is provided in Figure 14.

#### COCORD

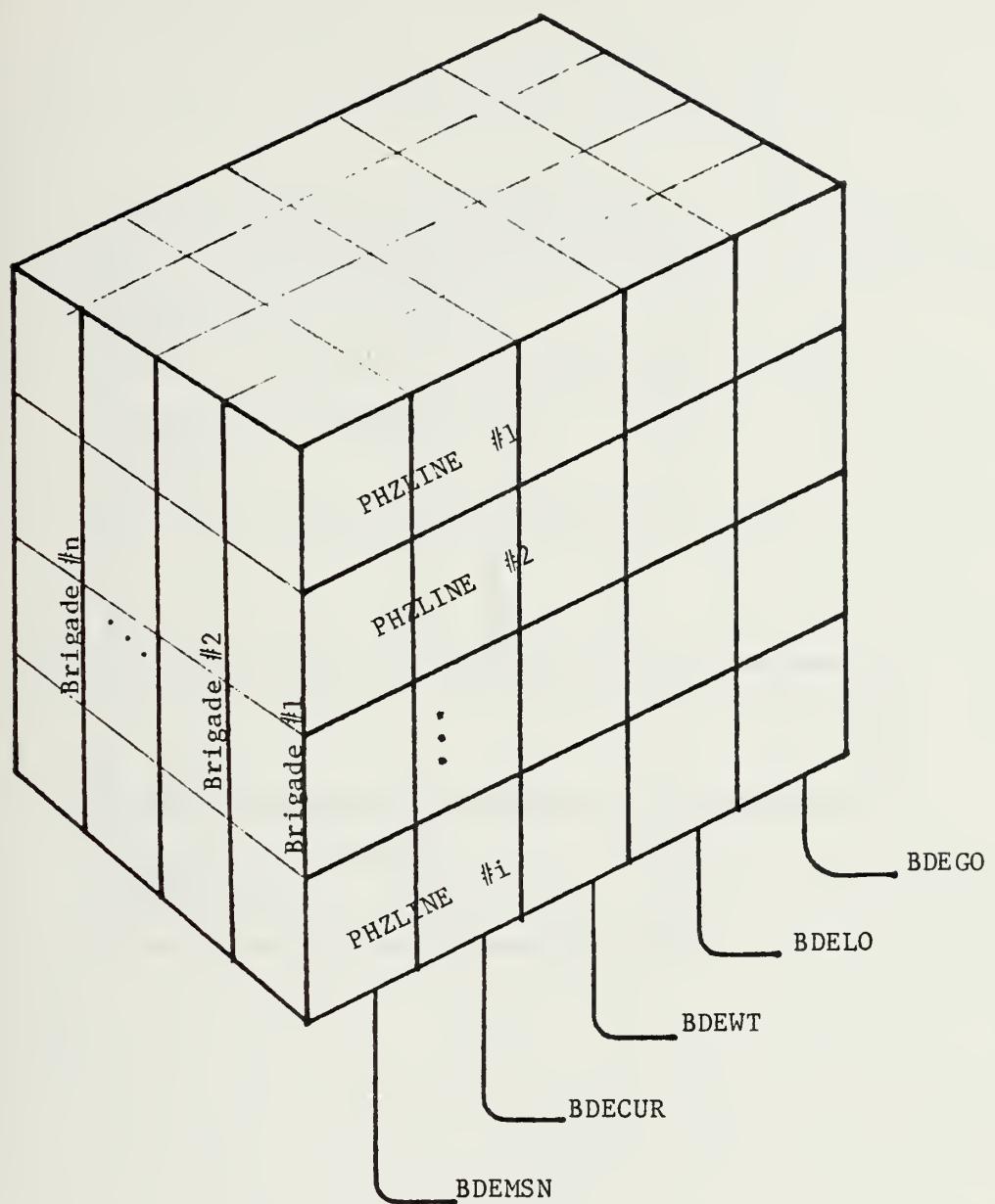
This array is the same one that is used by the production STAR model. It is explained in Appendix A. A diagram of the organization of the array is provided in Figure 12.

#### DIVCORD

This array is similar to the BNCORD AND BDECORD arrays. It contains the same type of data but for the division level as opposed to the brigade level. A diagram of the way this three dimensional array is



### BDECORD DIAGRAM



A plane for each brigade

A row in each plane for each phase line

A column in each plane for each of the five attributes

FIGURE 13



BNETTBL DIAGRAM

Subscripts of members of FMNET #1					
Subscripts of members of FMNET #2					
.	.	.			
Subscripts of members of FMNET #n					

FIGURE 14



organized is provided in Figure 15.

**RRPOINT**

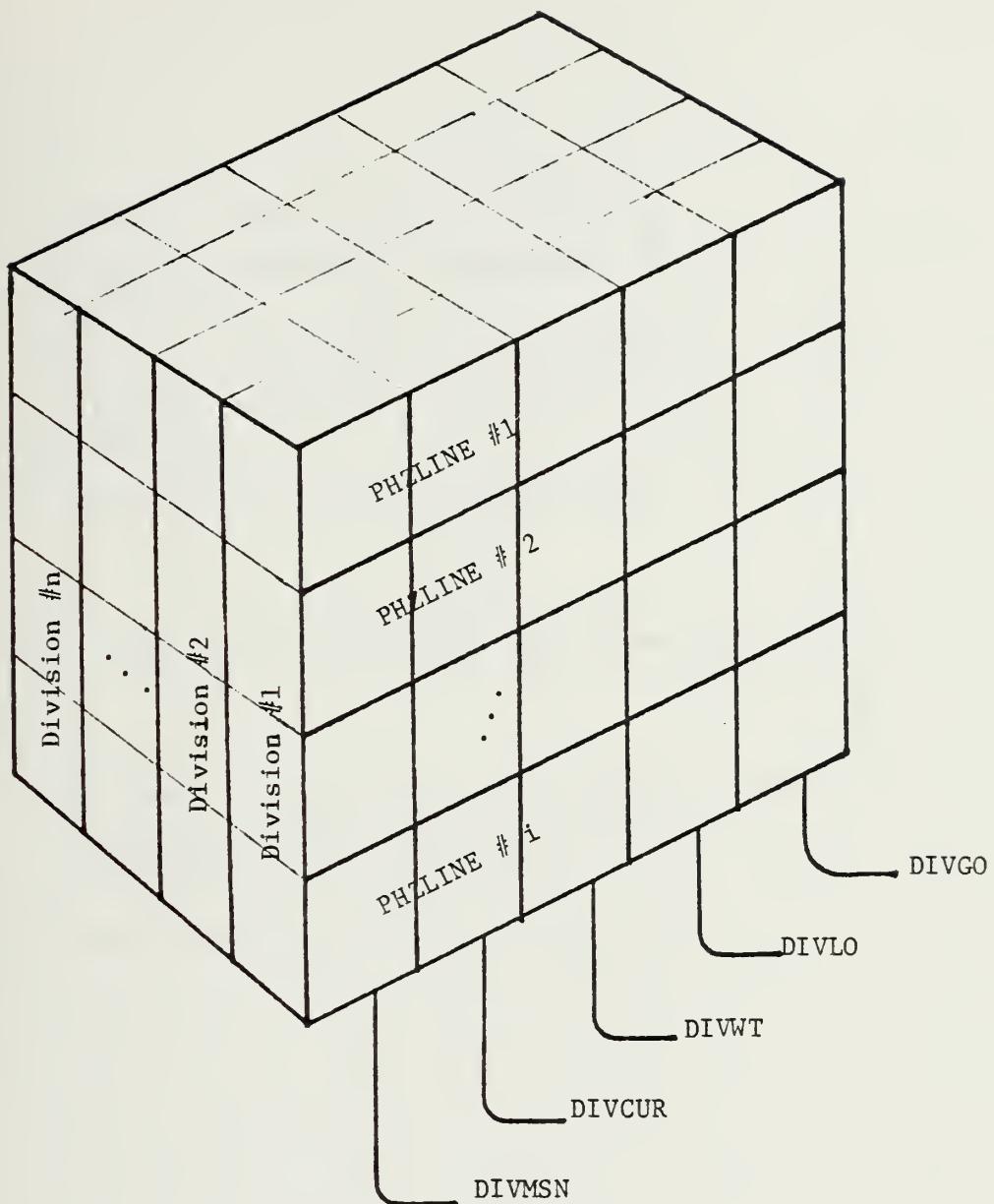
This two dimensional array is used primarily as an aid to debugging. It contains the pointer values of the UNITS/TANKS which belong to the  $i$ th radio net in the  $i$ th row of the array. This array is released after the initialization of the program where it is printed out since the pointers never change during the simulation. A diagram of the organization of this array is provided in Figure 16.

**TABLE**

This three dimensional array is the same as the one used by the production version of STAR. It is explained in Appendix A. A diagram of the organization of the array is provided in Figure 10.



### DIVCORD DIAGRAM



A plane for each division

A row in each plane for each phase line

A column in each plane for each of the five attributes

FIGURE 15



RRPOINT DIAGRAM

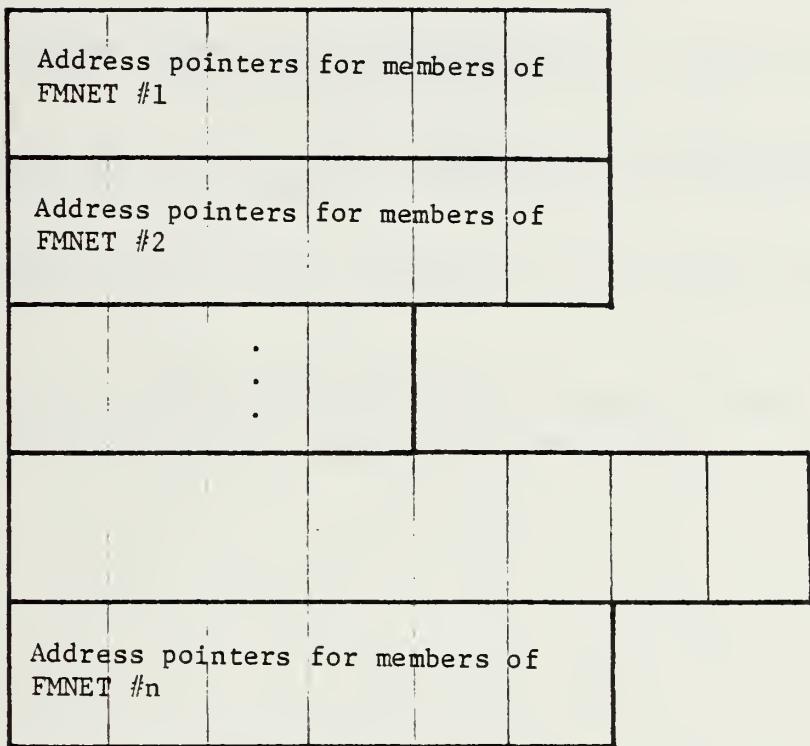


FIGURE 16



## APPENDIX G

### COMMUNICATIONS SETS

**Ceo.i.list**

This set contains pointers to all the SUBSCRIBERS who have access to a specific FMNET.

**Ether**

This set contains all the FMNET's which are simulated by the communications model.

**Mailbag**

This set contains pointers to all the MESSAGES generated by a given SUBSCRIBER for a given FMNET which haven't been sent.

**Rad.list**

This set contains pointers to all the RT.CONFIGS of a given TANK.

Figure 2 contains a complete diagram of all the sets and pointers used by the communications model.



## APPENDIX H

```

1 ROUTINE DECISION(ROW,COL,T)
2
3      ''THIS ROUTINE EXAMINES THE STATUS OF THE SECTION OR COMPANY TO WHICH
4          ''TANK T BELONGS AND DETERMINES WHETHER OR NOT MOVEMENT IS ALLOWED
5
6 IF COM.PRINT GT 20 PRINT 1 LINE WITH TIME.V THUS
7     INTO DECISION AT *****.***  

8 ALWAYS
9 LET ORDER= NO
10 IF MVARY(ROW,COL) GE 4 LET ORDER=YES
11     IF COM.PRINT GT 15
12         .     PRINT 1 LINE WITH ROW AND COL AS FOLLOWS
13 33333----STATUS OF MVARY(ROW=*****.COL=*****) CAUSED EXIT WITH ORDER=YES
14     SKIP 1 LINE
15 ALWAYS
16 GO TO ACT4, ACT5, ACT6 PER (MVARY(ROW,COL) - 3)
17 'ACT4'
18     CALL BN.GO(T,SYS,WPN)
19     GO TO ACT.NOW
20 'ACT5'
21     CALL OTHER.GO(T,SYS,WPN)
22     GO TO ACT.NOW
23 'ACT6'
24     CALL MOUNTER(T,SYS,WPN)
25 'ACT.NOW'
26     SCHEDULE A RE.MV.STATE(CG(T)) IN 60 UNITS
27 RETURN
28 ELSE
29 IF TABLE(PRTY(CG(T)),ROW,4)=0
30
31      ''THE SECTION OR COMPANY IS NOT RESTRICTED FROM MOVING AT WILL.
32
33 LET ORDER=YES LET FLAG=1
34 CALL EXIT(ORDER,T,CRIT,P.Z.TIME,FLAG)
35 LET ORIG=RRRPOINT(CG(T))
36 LET ADDR.STORE = RRRPOINT(NG.BN.UNIT(BN(T)))
37 CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
38
39      ''GENERATE A MESSAGE INFORMING THE BATTALION OF THE CG/SEC MOVE.
40
41 LET DEST.CODE(MSG) = BN.LEVEL
42 IF ROW EQ 7 LET MSG.TYPE(MSG) = CG.TELLS.BN.OF.CO.MOVE
43     CALL CG.GO(T,SYS,WPN)
44     SCHEDULE A RE.MV.STATE(CG(T)) IN 60 UNITS
45 IF RANGE(CHK.RNG) LE TABLE(PRTY(CG(T)),ROW,3)
46     PRINT 1 LINE AS FOLLOWS
47     *** DECISION BASED ON RANGE TO ENEMY *****
48     ALWAYS
49 ELSE
50     LET MSG.TYPE(MSG) = CG.NOTIF.BN.OF.SEC.MOVE

```



```

51      IF RANGE (CHK.RNG) LE TABLE (PRTY (C0 (T)), ROW, 3)
52          PRINT 1 LINE AS FOLLOWS
53      *** DECISION BASED ON RANGE TO ENEMY *****
54          CALL VEH.GO (T, TABLE (PRTY (C0 (T)), I, 1), TABLE (PRTY (C0 (T)), I, 2))
55          CALL CHG.STATUS (T)
56          RETURN
57      ELSE
58          LET SYS=TABLE (PRTY (C0 (T)), ROW, 1)
59          LET WPN=TABLE (PRTY (C0 (T)), ROW, 2)
60          CALL CHG.STATUS (T)
61          GO TO ACT1, ACT2 PER MVARY (ROW, COL)
62      'ACT1'
63          CALL SEC.GO (T, SYS, WPN)
64          GO TO ACT.LATER
65      'ACT2'
66          CALL PLT.GO (T, SYS, WPN)
67      'ACT.LATER'
68          SCHEDULE A RE.MV.STATE (C0 (T)) IN 60 UNITS
69          ALWAYS
70          SCHEDULE A COMMO.ATTEMPT (ORIG, MSG) NOW
71          RETURN
72      ELSE
73      IF CREQST (C0 (T)) EQ 1
74
75          ''THIS COMPANY HAS PREVIOUSLY REQUESTED PERMISSION TO MOVE.
76
77      IF COM.PRINT GT 15
78          PRINT 1 LINE AS FOLLOWS
79      11111-----CREQST (C0 (T)) EQ 1 CAUSED A RETURN
80          ALWAYS
81          RETURN
82      ELSE
83      LET ACHK=TRUNC.F (AREA.START (T) / 100)
84      IF ACHK GT PHAROW
85
86          ''DETERMINE IF THE COMPANY IS ON THE SAME PHASE LINE AS THE REST OF
87          ''ITS SISTER UNITS. IF IT IS FARTHER TO THE REAR, DO NOT ALLOW A
88          ''REQUEST TO MOVE.
89
90      IF COM.PRINT GT 15
91          PRINT 1 LINE WITH ACHK AND PHAROW AS FOLLOWS
92      22222-----STATUS OF ACHK=*** GT PHAROW=*** CAUSED A RETURN
93          ALWAYS
94          RETURN
95      ELSE
96      IF COCORD (C0 (T), ACHK, 1)=1
97
98          ''THE COMPANY IS RESTRICTED BUT HAS BEEN GIVEN PERMISSION TO MOVE.
99
100     LET ORDER=YES LET FLAG=2

```



```

101    CALL EXIT(ORDER,T,CRIT,P.Z.TIME,FLAG)
102    LET ORIG=RRRPOINT(CO(T))
103    LET ADDR.STORE = RRRPOINT(NG.BN.UNIT(BN(T)))
104    CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
105
106        ''GENERATE A MESSAGE INFORMING THE BATTALION OF THE CO/SEC MOVE.
107
108    LET DEST.CODE(MSG) = BN.LEVEL
109    IF ROW EQ 7 LET MSG.TYPE(MSG) = CO.TELLS.BN.OF.CO.MOVE
110        CALL CO.GO(T,STS,WPN)
111        SCHEDULE A RE.MV.STATE(CO(T)) IN 60 UNITS
112        IF RANGE(CHK.RNG) LE TABLE(PRTY(CO(T)),ROW,3)
113            PRINT 1 LINE AS FOLLOWS
114        *** DECISION BASED ON RANGE TO ENEMY *****
115        ALWAYS
116    ELSE
117        LET MSG.TYPE(MSG) = CO.NOTIF.BN.OF.SEC.MOVE
118        IF RANGE(CHK.RNG) LE TABLE(PRTY(CO(T)),ROW,3)
119            PRINT 1 LINE AS FOLLOWS
120        *** DECISION BASED ON RANGE TO ENEMY *****
121        CALL VEH.GO(T,TABLE(PRTY(CO(T)),I,1),TABLE(PRTY(CO(T)),I,2))
122        CALL CHG.STATUS(T)
123        RETURN
124    ELSE
125        LET SYS=TABLE(PRTY(CO(T)),ROW,1)
126        LET WPN=TABLE(PRTY(CO(T)),ROW,2)
127        CALL CHG.STATUS(T)
128        GO TO BCT1, BCT2 PER MVARY(ROW,COL)
129        'BCT1'
130            CALL SEC.GO(T,SYS,WPN)
131            GO TO BCT.LATER
132        'BCT2'
133            CALL PLT.GO(T,SYS,WPN)
134        'BCT.LATER'
135            SCHEDULE A RE.MV.STATE(CO(T)) IN 60 UNITS
136            ALWAYS
137            SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) NOW
138            RETURN
139    ELSE
140        LET FLAG=3
141        LET ORIG=RRRPOINT(CO(T))
142        LET ADDR.STORE = RRRPOINT(NG.BN.UNIT(BN(T)))
143        CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
144
145        ''GENERATE A MESSAGE TO THE BATTALION REQUESTING PERMISSION TO MOVE.
146
147    LET DEST.CODE(MSG) = BN.LEVEL
148    LET MSG.TYPE(MSG) = CO.BN.REQ.PERM.TO.MOVE
149    SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) NOW
150    RETURN

```



151 END

152

153



## APPENDIX I

```
1    ROUTINE DEC.CO
2    DEFINE J AS AN INTEGER VARIABLE
3
4    IF COM.PRINT GT 20 PRINT 1 LINE WITH TIME.V THUS
5        ENTERED DEC.CO ROUTINE AT TIME.V=*****.***.
6        ALWAYS
7        LET TEXT(*) = TEXT.POINTER
8        LET PHAROW = TEXT(1)
9        LET ORDER = TEXT(2)
10       LET T = TEXT(3)
11       LET CRIT = TEXT(4)
12       LET P.Z.TIME = TEXT(5)
13       LET TEXT(*)=0
14       IF CO.CAN.NOT.MOVE
15           LET COCORD(CO(T),PHAROW,1) = 0
16           IF COH.PRINT GT 15
17               PRINT 1 LINE WITH CO(T) THUS
18               COMPANY ** HAS BEEN ORDERED NOT TO MOVE
19               ALWAYS
20               SKIP 2 LINES
21               RETURN
22       ELSE
23       IF CO.MAY.MOVE
24           IF COM.PRINT GT 15
25               PRINT 1 LINE WITH CO(T) THUS
26               COMPANY ** HAS BEEN GIVEN PERMISSION TO MOVE AT WILL
27               SKIP 2 LINES
28               ALWAYS
29               LET COCORD(CO(T),PHAROW,1) = 1
30       ALWAYS
31       IF CO.ORDERED.TO.MOVE
32           IF COH.PRINT GT 15
33               PRINT 1 LINE WITH CO(T) THUS
34               COMPANY ** HAS BEEN ORDERED TO MOVE
35               SKIP 2 LINES
36               ALWAYS
37               LET COCORD(CO(T),PHAROW,1) = 1
38               FOR EACH TANK IN BLUE.ALIVE WITH CO(TANK) = COMPY(COMPANY.COMMANDER)
39                   UNTIL J=1, DO
40                   CALL AIR.COMMO(TANK,3)  LET CRIT=4
41                   CALL CO.GO(TANK,0,0)
42                   SCHEDULE A RE.MV.STATE(CO(TANK)) IN 60 UNITS
43                   LET J=J+1
44               LOOP
45               LET ORDER=2
46               LET FLAG = 4
47               CALL EXIT(ORDER,T,CRIT,P.Z.TIME,FLAG)
48               LET FLAG = 0
49               RETURN
50       ELSE
```



```
51 LET ACHK = TRUNC.F(AREA.START(T)/100)
52 IF COCORD(C0(T),ACHK,1)=1
53     LET ORDER=YES
54 ALWAYS
55 LET FLAG = 15
56 CALL EXIT(ORDER,T,CRIT,P.Z.TIME,FLAG)
57 LET FLAG = 0
58 RETURN
59 END
60
61
```



## APPENDIX J

```

1 ROUTINE DEC.BN
2
3 IF COM.PRINT GT 20 PRINT 1 LINE WITH TIME.V THUS
4     ENTERED DEC.BN ROUTINE AT TIME.V=xxxxxx.xxxx
5 ALWAYS
6 LET TEXT(1) = TEXT.POINTER
7 LET PHAROW = TEXT(1)
8 LET ORDER = TEXT(2)
9 LET T = TEXT(3)
10 LET CRIT = TEXT(4)
11 LET P.Z.TIME = TEXT(5)
12 LET TEXT(1)=0
13 IF CREQST(C0(T)) EQ 0 AND TYPE EQ CO.BN.REQ.PERM.TO.MOVE
14     LET CRIT=1 LET BNCUR(BN(T))=BNCUR(BN(T)) + COWT(C0(T))
15     LET CREQST(C0(T))=1
16 ALWAYS
17 IF CO.HAS.MOVED
18     IF COM.PRINT GT 15
19         PRINT 1 DOUBLE LINE WITH ORIG, ADDRSEE, AND TEXT.POINTER AS FOLLOWS
20 44444+++++CO xxxxxxxx NOTIFIED BN xxxxxxxx BY MESSAGE (TEXT=xxxxxxx) THAT IT IS
21 MOVING FROM ITS CURRENT POSITION.
22     ALWAYS
23     LET BNCUR(BN(T)) = BNCUR(BN(T)) + COWT(C0(T))
24     LET FLAG=16
25     LET T=RRRPOINT(NO.BN.UNIT(BN(T)))
26     LET ADDA.STORE = RRRPOINT(NO.BDE.UNIT(BDE(T)))
27     CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
28     LET DEST.CODE(MSG) = BDE.LEVEL
29     LET MSG.TYPE(MSG) = BN.NOTIF.BDE.8F.CO.MOVE
30     LET ORIG = T
31     SCHEDULE A COMM0.ATTEMPT(ORIG,MSG) IN COM06.DISTR(PAR60,PAR61,4) UNITS
32     RETURN
33 ELSE
34 IF SEC.HAS.MOVED
35     IF COM.PRINT GT 15
36         PRINT 1 DOUBLE LINE WITH ORIG, ADDRSEE, AND TEXT.POINTER THUS
37 66666+++++CO xxxxxxxx NOTIFIED BN xxxxxxxx BY MESSAGE (TEXT=xxxxxxx) THAT ONE O
38 F ITS SECTIONS IS MOVING FROM ITS POSITION.
39     ALWAYS
40     RETURN
41 ELSE
42 IF BN.ORDERED.TO.MOVE
43     LET BNCORD(BATT(BN(T)),PHAROW,1) = 1
44     LET BNCUR(BN(T))=BNCUR(BN(T))+BN0(BN(T))
45 ELSE
46     IF BN.MAY.MOVE
47         LET BNCORD(BATT(BN(T)),PHAROW,1) = 1
48         LET BNCUR(BN(T)) = BN0(BN(T))
49 ELSE
50     IF BN.CAN.NOT.MOVE

```



```

51           LET BNCORD(BATT(BN(T)),PHAROW,1) = 0
52           RETURN
53       ELSE
54           ALWAYS
55           ALWAYS
56           LET TEMP=ADRSEE LET ADRSEE=ORIG LET ORIG=TEMP
57           IF BNCORD(BATT(BN(T)),PHAROW,1) =1
58
59               **THIS BATTALION HAS PERMISSION TO MOVE AT WILL.
60
61           IF BNCRU(BN(T)) GE BNGO(BN(T))
62
63               **THE CRITICAL STATUS OF THE BATTALION HAS EXCEEDED ITS GO THRESHOLD.
64
65           IF COM.PRINT GT 15
66               PRINT 1 LINE WITH BN(T), TIME.V AS FOLLOWS
67               BN(*** NOT RESTRICTED AND ORDERED COMPANIES TO MOVE AT *****.***)
68               ALWAYS
69               FOR EACH COMPANY.COMMANDER IN BATTALION(BN(T)), DO
70                   LET FLAG=9
71                   LET T=RRRPOINT(COMPY(COMPANY.COMMANDER))
72                   LET ADDR.STORE = T
73                   CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME)
74                       YIELDING MSG
75                   LET DEST.CODE(MSG) = CO.LEVEL
76                   LET MSG.TYPE(MSG)=CO.TOLD.TO.MOVE.BY.BN
77                   SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) IN
78                       COMOS.DISTR(PAR58,PAR59,5) UNITS
79
80           LOOP
81           LET FLAG=15
82           LET T=RRRPOINT(NG.BN.UNIT(BN(T)))
83           LET ADDR.STORE = RRRPOINT(NG.BDE.UNIT(BDE(T)))
84           CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
85
86               **GENERATE A MESSAGE INFORMING BDE THAT ALL COS ARE BEING ORDERED TO
87               **MOVE AS SOON AS POSSIBLE.
88
89           LET DEST.CODE(MSG) = BDE.LEVEL
90           LET MSG.TYPE(MSG)= BN.TELLS.BDE.OF.BN.MOVE
91           IF COM.PRINT GT 15
92               PRINT 1 DOUBLE LINE WITH ORIG, ADDR.STORE, AND MSG.NO(MSG) THUS
93               77777*****BN ***** NOTIFIED BDE ***** BY MESSAGE NO ***** THAT IT HAS ORD
94               ERED ALL COMPANIES TO MOVE AS SOON AS POSSIBLE.
95               ALWAYS
96               SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) IN COMOS.DISTR(PAR60,PAR61,4) UNITS
97               RETURN
98           ELSE
99           IF BNCRU(BN(T)) GE BNLO(BN(T))
100              **THE BATTALION CRITICAL STATUS HAS EXCEEDED THE LO THRESHOLD.

```



```

101
102     IF COM.PRINT GT 15
103         PRINT 1 LINE WITH BN(T), TIME.V AS FOLLOWS
104         BN(*** NOT RESTRICTED AND GAVE COS PERMISSION TO MOVE AT *****.***)
105         ALWAYS
106             FOR EACH COMPANY.COMMANDER IN BATTALION(BN(T)), DO
107                 LET FLAG=10
108                 LET T=RRRPOINT(COMPY(COMPANY.COMMANDER))
109                 LET ADDR.STORE = T
110                 CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME)
111                     YIELDING MSG
112                 LET DEST.CODE(MSG) = CO.LEVEL
113                 LET MSG.TYPE(MSG) = BN.GAVE.CO.PERM.TO.MOVE
114                 SCHEDULE A COMM0.ATTEMPT(ORIG,MSG) IN
115                     COM05.DISTR(PAR58,PAR59,5) UNITS
116             LOOP
117             LET FLAG=11
118             LET T=RRRPOINT(NO.BN.UNIT(BN(T)))
119             LET ADDR.STORE = RRPPOINT(NO.BDE.UNIT(BDE(T)))
120             CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
121
122             **GENERATE A MESSAGE INFORMING BDE THAT ALL COS HAVE BEEN GIVEN
123             **PERMISSION TO MOVE AT WILL.
124
125             LET DEST.CODE(MSG) = BDE.LEVEL
126             LET MSG.TYPE(MSG) = BN.TELLS.BDE.OF.BN.MOVE
127             IF COM.PRINT GT 15
128                 PRINT 1 DOUBLE LINE WITH ORIG, ADDR.STORE, AND MSG.NO(MSG) THUS
129                 88888*****BN ***** NOTIFIED BDE ***** BY MESSAGE NO ***** THAT IT HAS GIV
130                 EN ALL COMPANIES PERMISSION TO MOVE AT WILL.
131             ALWAYS
132             SCHEDULE A COMM0.ATTEMPT(ORIG,MSG) IN COM06.DISTR(PAR60,PAR61,4) UNITS
133             RETURN
134
135             ELSE
136                 IF COM.PRINT GT 15
137                     PRINT 1 LINE WITH BN(T), TIME.V AS FOLLOWS
138                     BN(*** NOT RESTRICTED AND ORDERED COMPANY NOT TO MOVE AT *****.***)
139                     ALWAYS
140                         LET FLAG = 12
141                         LET ADDR.STORE = ADRSEE
142                         LET T = ADRSEE
143                         CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
144                         LET DEST.CODE(MSG) = CO.LEVEL
145                         LET MSG.TYPE(MSG) = BN.DIR.CO.NOT.TO.MOVE
146                         SCHEDULE A COMM0.ATTEMPT(ORIG,MSG) IN COM05.DISTR(PAR58,PAR59,5) UNITS
147                         RETURN
148             ELSE
149                 IF BNCUR(BN(T)) GE BNLC(BN(T))
150                     IF COM.PRINT GT 15
151                         PRINT 1 LINE WITH BN(T) AND TIME.V THUS

```



```
151      BN(XX) IS RESTRICTED AND IS REQUESTING PERMISSION TO MOVE AT *****.***  
152      ALWAYS  
153      LET FLAG = 13  
154      LET T=RRRPOINT (NO.BN.UNIT (BN(T)))  
155      LET ADDR.STORE = RRRPOINT (NO.BDE.UNIT (BDE(T)))  
156      CALL GEN.MOVE.DECISION.MSG (PHAROW, ORDER, T, CRIT, P.Z.TIME) YIELDING MSG  
157      LET DEST.CODE (MSG) = BDE.LEVEL  
158      LET MSG.TYPE (MSG) = BN.BDE.REQ.PERM.TO.MOVE  
159      SCHEDULE A COMMO.ATTEMPT (ORIG,MSG) IN COMOS.DISTR (PAR60,PAR61,4) UNITS  
160      RETURN  
161  ELSE  
162      IF COM.PRINT GT 15  
163          PRINT 1 LINE WITH BN(T) AND TIME.V THUS  
164          BN(XX) IS RESTRICTED AND ORDERED THE REQUESTING CO NOT TO MOVE AT *****.***  
165          ALWAYS  
166          LET FLAG=14  
167          CALL EXIT (ORDER,T,CRIT,P.Z.TIME,FLAG)  
168          LET ADDR.STORE = ADDRSEE  
169          LET T = ADDRSEE  
170          CALL GEN.MOVE.DECISION.MSG (PHAROW, ORDER, T, CRIT, P.Z.TIME) YIELDING MSG  
171          LET DEST.CODE (MSG) = CO.LEVEL  
172          LET MSG.TYPE (MSG) = BN.DIR.CO.NOT.TO.MOVE  
173          SCHEDULE A COMMO.ATTEMPT (ORIG,MSG) IN COMOS.DISTR (PAR58,PAR59,5) UNITS  
174          RETURN  
175  ENO  
176  
177
```



## APPENDIX K

```

1   ROUTINE DEC.BOE
2
3   IF COM.PRINT GT 20 PRINT 1 LINE WITH TIME.V THUS
4       ENTERED DEC.BOE ROUTINE AT TIME.V=*****.****
5   ALWAYS
6   LET TEXT(*) = TEXT.POINTER
7   LET PHAROW = TEXT(1)
8   LET ORDER = TEXT(2)
9   LET T = TEXT(3)
10  LET CRIT = TEXT(4)
11  LET P.Z.TIME = TEXT(5)
12  LET TEXT(*)=0
13  IF BREQST(BN(T))=0 AND TYPE EQ BN.BOE.REQ.PERM.TG.MOVE
14      LET CRIT=2
15      LET BOECUR(BDE(T)) = BDECUR(BDE(T)) + BNWT(BN(T))
16      LET BREQST(BN(T))=1
17   ALWAYS
18   IF BN.HAS.MOVED
19       IF COM.PRINT GT 15
20           PRINT 1 DOUBLE LINE WITH ORIG. ADDRESSEE(MSG), AND TEXT.POINTER THUS
21           55555+***BN ***** NOTIFIED BRIGADE ***** BY MESSAGE ***** THAT IT HAS
22           MOVED FROM ITS POSITION
23       ALWAYS
24       LET BOECUR(BDE(T)) = BDECUR(BDE(T)) + BNWT(BN(T))
25       LET FLAG=21
26       LET T=RRRPOINT(NG.BOE.UNIT(BDE(T)))
27       LET ADDR.STORE = RRRPOINT(NG.DIV.UNIT(DIV(T)))
28       CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
29       LET DEST.CODE(MSG) = DIV.LEVEL
30       LET MSG.TYPE(MSG) = BOE.NOTIF.DIV.OF.BN.MOVE
31       LET ORIG = T
32       SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) IN COMO3.DISTR(PAR54,PAR55,9) UNITS
33       RETURN
34   ELSE
35   IF CO.HAS.MOVED
36       IF COM.PRINT GT 15
37           PRINT 1 DOUBLE LINE WITH ORIG, ADSEE, AND TEXT.POINTER AS FOLLOWS
38           AAAA+***BN ***** NOTIFIED BOE ***** BY MESSAGE (TEXT=*****) THAT ONE
39           OF ITS COMPANIES IS MOVING FROM ITS POSITION.
40       ALWAYS
41       RETURN
42   ELSE
43   IF BDE.ORDERED.TG.MOVE
44       LET BDECORD(BDE(BDE(T)),PHAROW,1) = 1
45       LET BOECUR(BDE(T)) = BDECUR(BDE(T)) + BDEGO(BDE(T))
46   ELSE
47       IF BDE.MAY.MOVE
48           LET BDECORD(BDE(BDE(T)),PHAROW,1) = 1
49           LET BOECUR(BDE(T)) = BDEGO(BDE(T))
50   ELSE

```



```

51      IF BDE.CAN.NOT.MOVE
52          LET BDECORD(BDE(BDE(T)),PHAROW,1) = 0
53          RETURN
54      ELSE
55          ALWAYS
56          ALWAYS
57          LET TEMP=ADRSEE LET ADRSEE=ORIG LET ORIG=TEMP
58          IF BDECORD(BDE(BDE(T)),PHAROW,1) = 1
59
60              ''THIS BRIGADE HAS PERMISSION TO MOVE AT WILL.
61
62      IF BDECUR(BDE(T)) GE BDEGO(BDE(T))
63
64          ''THE CRITICAL STATUS OF THE BRIGADE HAS EXCEEDED ITS GO THRESHOLD.
65
66          LET CRIT = 4
67          IF COM.PRINT GT 15
68              PRINT 1 LINE WITH BDE(T) AND TIME.V THUS
69              BDE(*** NOT RESTRICTED AND ORDERED BATTALIONS TO MOVE AT *****.***)
70          ALWAYS
71          FOR EACH BN.COMMANDER IN BRIGADE(BDE(T)),DO
72              LET FLAG=5
73              LET T=ARRPOINT(NO.BN.UNIT(BN.COMMANDER))
74              LET ADDR.STORE = T
75              CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
76              LET DEST.CODE(MSG) = BN.LEVEL
77              LET MSG.TYPE(MSG)=BN.TOLD.TO.MOVE.BY.BDE
78              SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) IN COMO4.DISTR(PAR56,PAR57,7) UNITS
79          LOOP
80          LET FLAG = 17
81          LET T = ARRPOINT(NO.BDE.UNIT(BDE(T)))
82          LET ADDR.STORE = ARRPOINT(NO.DIV.UNIT(BDE(T)))
83          CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
84
85          ''GENERATE A MESSAGE INFORMING DIV THAT ALL BNS ARE BEING ORDERED TO
86          ''MOVE AS SOON AS POSSIBLE.
87
88          LET DEST.CODE(MSG) = DIV.LEVEL
89          LET MSG.TYPE(MSG) = BDE.LETS.DIV.KNOW.OF.BDE.MOVE
90          IF COM.PRINT GT 15
91              PRINT 1 DOUBLE LINE WITH ORIG, ADDR.STORE, AND MSG.NO(MSG) THUS
92              BBBBBB+++++BDE ***** NOTIFYING DIV ***** BY MESSAGE NO ***** THAT IT HAS OR
93              DERED ALL BATTALIONS TO MOVE AS SOON AS POSSIBLE.
94          ALWAYS
95          SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) IN COMO3.DISTR(PAR54,PAR55,9) UNITS
96          RETURN
97      ELSE
98      IF BDECUR(BDE(T)) GE BDELO(BDE(T))
99
100         ''THE BRIGADE CRITICAL STATUS HAS EXCEEDED THE LO THRESHOLD.

```



```

101      LET CRIT=3
102      IF COM.PRINT GT 15
103          PRINT 1 LINE WITH BOE(T) AND TIME.V THUS
104          BOE(XX) NOT RESTRICTED AND GAVE BNS PERMISSION TO MOVE AT *****.*****
105          ALWAYS
106          FOR EACH BN.COMMANDER IN BRIGADE(BOE(T)), DO
107              LET FLAG=6
108              LET T=RRRPOINT(NO.BN.UNIT(BN.COMMANDER))
109              LET ADDR.STORE = T
110              CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
111              LET DEST.CODE(MSG) = BN.LEVEL
112              LET MSG.TYPE(MSG)=BOE.GAVE.BN.PERM.TO.MOVE
113              SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) IN COMO4.DISTR(PAR56,PAR57,7) UNITS
114              LOOP
115                  LET FLAG = 18
116                  LET T = RRPPOINT(NO.BDE.UNIT(BDE(T)))
117                  LET ADDR.STORE = RRPPOINT(NO.DIV.UNIT(BDE(T)))
118                  CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
119
120                  ''GENERATE A MESSAGE INFORMING DIV THAT ALL BNS HAVE BEEN GIVEN
121                  ''PERMISSION TO MOVE AT WILL.
122
123
124                  LET DEST.CODE(MSG) = DIV.LEVEL
125                  LET MSG.TYPE(MSG) = BDE.LETS.DIV.KNOW.OF.BDE.MOVE
126                  IF COM.PRINT GT 15
127                      PRINT 1 DOUBLE LINE WITH ORIG, ADDR.STORE, AND MSG.NO(MSG) THUS
128                      CCCCC+++++BOE ***** NOTIFIED DIV ***** BY MESSAGE NO ***** THAT IT HAS GI
129                      YEN ALL BATTALIONS PERMISSION TO MOVE AT WILL.
130                      ALWAYS
131                      SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) IN COMO3.DISTR(PAR54,PAR55,9) UNITS
132                      RETURN
133
134      ELSE
135          IF COM.PRINT GT 15
136              PRINT 1 LINE WITH BOE(T) AND TIME.V AS FOLLOWS
137              BOE(XX) NOT RESTRICTED AND ORDERED THE BATTALION NOT TO MOVE AT *****.*****
138              ALWAYS
139              LET FLAG=7
140              LET ADDR.STORE = ADSEE
141              LET T = ADSEE
142              CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
143              LET DEST.CODE(MSG) = BN.LEVEL
144              LET MSG.TYPE(MSG)=BDE.DIR.BN.NOT.TO.MOVE
145              SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) IN COMO4.DISTR(PAR56,PAR57,7) UNITS
146              RETURN
147
148      ELSE
149          IF BDECUR(BOE(T)) GE BDEL0(BOE(T))
150              IF COM.PRINT GT 15
151                  PRINT 1 LINE WITH BOE(T) AND TIME.V THUS
152                  BOE(XX) IS RESTRICTED AND IS REQUESTING PERMISSION TO MOVE AT *****.*****

```



```
151      ALWAYS
152      LET FLAG = 19
153      LET T=RRRPOINT (NO.BOE.UNIT (BOE (T)))
154      LET ADDR.STORE = RRPPOINT (NO.DIV.UNIT (DIV (T)))
155      CALL GEN.MOVE.DECISION.MSG (PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
156      LET DEST.CODE (MSG) = DIV.LEVEL
157      LET MSG.TYPE (MSG) = BOE.DIV.REQ.PERM.TO.MOVE
158      SCHEDULE A COMMO.ATTEMPT (ORIG,MSG) IN COMO3.DISTR (PAR54,PAR55,9) UNITS
159      RETURN
160  ELSE
161      IF COM.PRINT GT 15
162          PRINT 1 LINE WITH BOE (T) AND TIME.V THUS
163          BOE (**) IS RESTRICTED AND ORDERED THE REQUESTING BN NOT TO MOVE AT ****.**
164          ALWAYS
165      LET FLAG = 20
166      CALL EXIT (ORDER,T,CRIT,P.Z.TIME,FLAG)
167      LET ADDR.STORE = ADRSEE
168      LET T = ADRSEE
169      CALL GEN.MOVE.DECISION.MSG (PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
170      LET DEST.CODE (MSG) = BN.LEVEL
171      LET MSG.TYPE (MSG) = BOE.DIR.BN.NOT.TO.MOVE
172      SCHEDULE A COMMO.ATTEMPT (ORIG,MSG) IN COMO4.DISTR (PAR56,PAR57,7) UNITS
173      RETURN
174  END
175
176
```



## APPENDIX L

```
1 ROUTINE DEC.DIV
2
3 IF COM.PRINT GT 20
4     PRINT 1 LINE WITH TIME.V THUS
5     ENTERED DEC.DIV ROUTINE AT TIME.V = *****.****
6 ALWAYS
7 LET TEXT(*) = TEXT.POINTER
8 LET PHAROW = TEXT(1)
9 LET ORDER = TEXT(2)
10 LET T = TEXT(3)
11 LET CRIT = TEXT(4)
12 LET P.Z.TIME = TEXT(5)
13 LET TEXT(*)=0
14 IF BDEREQUEST(BDE(T))=0 AND TYPE EQ BDE.DIV.REQ.PERM.TO.MOVE
15     LET CRIT=5
16     LET DIVCUR(DIV(T))=DIVCUR(DIV(T)) + BDEWT(BDE(T))
17     LET DIVREQUEST(BDE(T))=1
18 ALWAYS
19     IF COM.PRINT GT 15
20         PRINT 1 LINE WITH DIV(T) AND TIME.V THUS
21         DIV(***) NOT RESTRICTED AND GAVE BDES PERMISSION TO MOVE AT ****.****
22     ALWAYS
23     FOR EACH BDE.COMMANDER IN DIVISION(DIV(T)), DO
24         LET FLAG=22
25         LET T=RRRPOINT(NO.BDE.UNIT(BDE.COMMANDER))
26         LET ADDR.STORE = T
27         CALL GEN.MOVE.DECISION.MSG(PHAROW,ORDER,T,CRIT,P.Z.TIME) YIELDING MSG
28         LET DEST.CODE(MSG) = BDE.LEVEL
29         LET MSG.TYPE(MSG)=DIV.GAVE.BDE.PERM.TO.MOVE
30         SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) IN COM20.DISTR(PAR71,PAR72,7) UNITS
31     LOOP
32 RETURN
33 END
34
35
```



## APPENDIX M

```
1 ROUTINE EXIT(ORDER,T,CRIT,P.Z.TIME,FLAG)
2
3 IF CRIT NE 0 AND CRIT LE 3
4     LET CRIT = CRIT - 1
5     CALL AIR.COMM0(T,CRIT)
6     LET CRIT = CRIT + 1
7 ALWAYS
8 IF (ORDER=0 OR ORDER=2) AND P.Z.TIME=NO
9     SCHEDULE A PHAZ.CMK IN 60 UNITS LET P.Z.TIME=YES
10 ALWAYS
11 IF ORDER=2 LET ORDER=NO
12 ALWAYS
13 IF COM.PRINT GT 20 PRINT 1 LINE WITH TIME.V AND FLAG THUS
14     OUT OF DECISION AT *****.*** THROUGH EXIT ***
15 ALWAYS
16 RETURN
17 END
18
19
```



## APPENDIX N

```
1 ROUTINE GEN.MOVE.DECISION.MSG (PHAROW, ORDER, T, CRIT, P.Z.TIME) YIELDING MSG
2
3 CREATE A MESSAGE CALLED MSG
4 RESERVE TEXT(1) AS 5
5 LET MSG.TEXT(MSG)=TEXT(1)
6 LET TEXT(1)=PHAROW
7 LET TEXT(2)=ORDER
8 LET TEXT(3)=T
9 LET TEXT(4)=CRIT
10 LET TEXT(5)=P.Z.TIME
11 LET ADDRESSEE(MSG)=ADDR.STORE
12 LET LENGTH.MSG(MSG)=30.0
13 LET LST.SEND.TIME(MSG)=TIME.V + 1200.0
14 LET MSG.NO(MSG)=NXT.MSG.NO(FMTANK.LIST(ORIG))
15 ADD 1 TO NXT.MSG.NO(FMTANK.LIST(ORIG))
16 IF COM.PRINT GT 15
17     SKIP 1 LINE
18     PRINT 1 LINE WITH TIME.V AS FOLLOWS
19     ****EXECUTING GEN.MOVE.DECISION.MSG ROUTINE AT TIME.V*****.****
20     PRINT 1 DOUBLE LINE WITH MSG.NO(MSG), ORIG, ADDRESSEE(MSG), TEXT(1),
21             PHAROW, ORDER, T, CRIT, AND FLAG THUS
22     MSG.NO ***** FROM ***** TO ***** TEXT ***** PHAROW ** ORDER
23     ** T ***** CRIT ** RETURN **
24 ALWAYS
25 LET TEXT(1)=0
26 LET FLAG=0
27 RETURN
28 END
29
30
```



## APPENDIX O

```
1
2
3 ROUTINE COM.MAIN
4 RESERVE DUMMY(%) AS 1
5 CALL BUILDNETS
6 CALL CHECKNETS
7 CALL BNNTBL
8 CALL COMMGEAR
9 CALL COMMCHECK
10 CALL INITIALIZE
11 LET FMNET = F.ETHER
12 SCHEDULE A DUMP.MAILBAG IN 600.0 UNITS
13 SCHEDULE A MSG.GEN(FMNET) IN COM01.DISTR(PAR50,PAR51,1) UNITS
14 IF COM.PRINT GT 25
15     START NEW PAGE
16     LIST ATTRIBUTES OF EACH B.MVR.CDR
17     IF LINE.V GT 20
18         START NEW PAGE
19         ALWAYS
20             LIST ATTRIBUTES OF EACH UNIT IN BLUE.ALIVE
21         ALWAYS
22         START NEW PAGE
23     RETURN
24 END
```



## APPENDIX P

```

1      ROUTINE BUILDNETS
2      DEFINE I, J, K, NALTFR, NETNO, NFMN, NONETS, NORELAY, NPRI, NRSUB, NTCK, TKALT,
3          AND TKHOLD AS INTEGER VARIABLES
4
5      READ NONETS, NORELAY
6      IF NONETS IS EQUAL TO ZERO,
7          GO TO X3
8      OTHERWISE
9          FOR I = 1 TO NONETS, DO
10             LET NETNO = 0  "DONE TO AID ERROR IDENTIFICATION"
11             READ NFMN, NRSUB, NALTFR
12             READ NTCK AND NPRI
13             IF NTCK IS NOT EQUAL TO NFMN,
14                 GO TO ERROR
15             OTHERWISE
16                 CREATE AN FMNET
17                 FILE FMNET IN THE ETHER
18                 FOR J=1 TO NRSUB, DO
19                     READ TKHOLD
20                     CREATE A SUBSCRIBER
21                     LET TFC.WAITING(SUBSCRIBER) = 0
22                     LET OWN.NET(SUBSCRIBER) = FMNET
23                     LET OWN.TANK(SUBSCRIBER) = RRPPOINT(TKHOLD)
24                     FILE SUBSCRIBER IN CEO1.LIST(FMNET)
25                     LOOP
26                     LET NUM.NET(FMNET) = NFMN
27                     LET FM.IDLE(FMNET) = IDLE
28                     LET FMPRI.FREQ(FMNET) = NPRI
29                     IF NALTFR IS EQUAL TO ZERO,
30                         LET FMALT.FREQ(FMNET) = 0
31                         GO TO X4
32                     OTHERWISE
33                         READ NETNO
34                         IF NETNO IS NOT EQUAL TO NFMN,
35                             GO TO ERROR
36                         OTHERWISE
37                         RESERVE FM.PUSHES(1) AS (NALTFR + 1)
38                         LET FMALT.FREQ(FMNET) = FM.PUSHES(1)
39                         LET FM.PUSHES(1) = NALTFR
40                         FOR K=1 TO NALTFR, DO
41                             READ TKALT
42                             LET FM.PUSHES(K+1) = TKALT
43                             LOOP
44             'X4'  "CONTINUE"
45             LET FM.PUSHES(1) = 0
46             LOOP
47             'X3'  "CONTINUE"
48             RETURN
49
50     'ERROR'  "CONTINUE"

```



```
51      PRINT 2 LINES WITH NFMN, NTCK AND NETNO LIKE THIS
52      ERROR, NET NUMBER *****, SUBS. NUMBER *****,  

53      AND ALT.FREQ NET NUMBER ***** ARE INCONSISTENT
54          SKIP 1 LINE
55          PRINT 1 LINE WITH NONETS AND NORELAY LIKE THIS
56          NONETS=***** , NORELAY=*****
57          PRINT 1 LINE WITH NFMN, NRSUB, NALTFR LIKE THIS
58          NFMN=***** , NRSUB=***** , NALTFR=*****
59          PRINT 1 LINE WITH NTCK,NPRI,TKHOLD LIKE THIS
60          NTCK=***** , NPRI=***** , TKHOLD= *****
61          STOP
62      END
63
64
```



## APPENDIX Q

```
1 ROUTINE CHECKNETS
2 DEFINE I AND NFMALT AS INTEGER VARIABLES
3
4 FOR EACH FMNET IN THE ETHER, DO
5   IF LINE.V GT 57 START NEW PAGE
6   ALWAYS
7     PRINT 1 LINE WITH NUM.NET(FMNET) LIKE THIS
8     FOR NET NUMBER *****
9     SKIP 1 LINE
10    PRINT 1 LINE WITH FMPRI.FREQ(FMNET) LIKE THIS
11    PRIMARY FREQUENCY IS ***** KHZ
12
13  IF FMALT.FREQ(FMNET) IS NOT EQUAL TO ZERO.
14    LET FM.PUSHES(0) = FMALT.FREQ(FMNET)
15    LET NFMALT = FM.PUSHES(0)
16    PRINT 1 LINE WITH NFMALT LIKE THIS
17    THERE ARE ***** ALTERNATE FREQUENCIES
18    FOR I = 1 TO NFMALT. DO
19      PRINT 1 LINE WITH FM.PUSHES(I+1) LIKE THIS
20      *****
21      LOOP
22  OTHERWISE
23    PRINT 1 LINE LIKE THIS
24    THERE ARE NO ALTERNATE FREQUENCIES
25  ALWAYS
26  PRINT 1 LINE WITH N.CEOI.LIST(FMNET) LIKE THIS
27  THERE ARE ***** SUBSCRIBERS IN THE NET,
28  SKIP 3 LINES
29  LOOP
30  RETURN
31  END
32
33
```



## APPENDIX R

```
1 ROUTINE BNNTBL
2 DEFINE I AND J AS INTEGER VARIABLES
3
4 RESERVE NETTBL(*,*) AS N.ETHER BY *
5 RESERVE RPOINT(*,*) AS N.ETHER BY *
6 LET I = 1
7 FOR EVERY FMNET IN THE ETHER, DO
8     RESERVE NETTBL(I,*) AS N.CEOI.LIST(FMNET)
9     RESERVE RPOINT(I,*) AS N.CEOI.LIST(FMNET)
10    LET J = 1
11    FOR EVERY SUBSCRIBER IN CEOI.LIST(FMNET), DO
12        LET NETTBL(I,J) = TK.ID(OWN.TANK(SUBSCRIBER))
13        LET RPOINT(I,J) = SUBSCRIBER
14        LET J = J + 1
15    LOOP
16    LET I = I + 1
17 LOOP
18
19 IF LINE.V GT 62 START NEW PAGE
20 ALWAYS
21 PRINT 1 LINE LIKE THIS
22     THE ITH ROW OF ARRAY NETTBL CONTAINS THE SUBSCRIBERS IN THE ITH RADIO NET
23 LIST NETTBL
24 SKIP 2 LINES
25 IF LINE.V GT 68 START NEW PAGE
26 . ALWAYS
27 PRINT 1 DOUBLE LINE LIKE THIS
28     THE ITH ROW OF THE ARRAY RPOINT CONTAINS THE ADDRESS POINTERS OF THE SUBSCR
29 IBERS IN THE ITH RADIO NET
30 LIST RPOINT
31 SKIP 2 LINES
32 RELEASE RPOINT(*,*)
33 RETURN
34 END
35
36
```



## APPENDIX S

```
1 ROUTINE COMMGEAR
2 DEFINE I, J, NORAD, NRADIO.TANKS, AND TKNO AS INTEGER VARIABLES
3
4 READ NRADIO.TANKS
5
6 FOR I = 1 TO NRADIO.TANKS, DO
7   READ TKNO, NORAD
8   CREATE A MYSET
9   LET TANK = RRPPOINT(TKNO)
10  LET FMTANK.LIST(TANK) = MYSET
11  FOR J = 1 TO NORAD, DO
12    CREATE AN RT.CONFIG
13    FILE RT.CONFIG IN RAD.LIST(MYSET)
14    READ RAD.TYPE(RT.CONFIG) AND MODE.OPERATION(RT.CONFIG)
15    READ ANT.TYPE(RT.CONFIG)
16    IF ANT.TYPE(RT.CONFIG) IS LESS THAN ZERO,
17      "DIRECTIONAL ANTENNA INPUT"
18      PRINT 3 LINES WITH TKNO, NORAD, J, RAD.TYPE(RT.CONFIG),
19      MODE.OPERATION(RT.CONFIG) AND ANT.TYPE(RT.CONFIG) LIKE THIS
20      ON TANK *****, WITH A TOTAL OF **** RADIOSETS, FOR RADIO SET
21      NUMBER *****, OF TYPE *****, MODE *****, AND ANTENNA *****
22      ----THIS SET OPERATES A DIRECTIONAL ANTENNA
23      STOP
24      ALWAYS
25      READ REM.ANT(RT.CONFIG)
26      IF REM.ANT(RT.CONFIG) IS NOT EQUAL TO ZERO,
27        "REMOTED ANTENNA INPUT"
28        PRINT 3 LINES WITH TKNO, NORAD, J, RAD.TYPE(RT.CONFIG),
29        MODE.OPERATION(RT.CONFIG) AND ANT.TYPE(RT.CONFIG) LIKE THIS
30        ON TANK *****, WITH A TOTAL OF **** RADIOSETS, FOR RADIO SET
31        NUMBER *****, OF TYPE *****, MODE *****, AND ANTENNA *****
32        ----THIS SET OPERATES WITH A REMOTED ANTENNA
33        STOP
34        REGARDLESS
35      LOOP
36    LOOP
37
38    "USE NET STRUCTURE INPUT PREVIOUSLY TO ASSIGN EQUIPMENTS"
39    "TO NETS"
40
41    IF LINE.V GT 30 START NEW PAGE
42    ALWAYS
43    PRINT 4 LINES AS FOLLOWS
44    RRPPOINT(TK.ID(TANK)) = TANK
45    FMTANK.LIST(TANK) = MYSET(TANK)
46
47
48    FOR EVERY FMNET IN THE ETHER, DO
49    PRINT 1 LINE WITH NUM.NET(FMNET) AND FMNET AS FOLLOWS
50    NUM.NET(FMNET) = ***    FMNET POINTER = *****
```



```

51      SKIP 1 LINE
52      FOR EVERY SUBSCRIBER IN CEOI.LIST(FMNET) , DO
53          LET TANK = OWN.TANK(SUBSCRIBER)
54          IF FMTANK.LIST(TANK) EQUALS ZERO,
55              GO TO ERROR1
56          OTHERWISE
57          LET MYSET = FMTANK.LIST(TANK)
58          LET NXT.MSG.NO(MYSET) = (TK.ID(OWN.TANK(SUBSCRIBER)) * 1000) + 1
59          FOR EVERY RT.CONFIG IN RAD.LIST(MYSET) , DO
60              IF OWN.SUBSCRIBER(RT.CONFIG) EQUALS ZERO,
61                  GO TO JOIN.NET
62              OTHERWISE
63              LOOP
64              GO TO ERROR2
65
66      'JOIN.NET'  ''PLACE THIS EQUIPMENT ON THE NET''
67          LET OWN.SUBSCRIBER(RT.CONFIG) = SUBSCRIBER
68          PRINT 1 DOUBLE LINE WITH TK.ID(TANK) , TANK, FMTANK.LIST(TANK) , RT.CONFIG,
69          AND OWN.SUBSCRIBER(RT.CONFIG) AS FOLLOWS
70          TK.ID(TANK)=*****          TANK=*****          FMTANK.LIST(TANK)=*****
71          RT.CONFIG=*****          SUBSCRIBER=*****
72
73          LOOP
74          SKIP 1 LINE
75          LOOP
76          SKIP 5 LINES
77
78      RETURN
79
80      'ERROR1'  ''ERROR WHEN TANK HAS NO RADIOS''
81          PRINT 1 LINE WITH TK.ID(TANK) AND NUM.NET(FMNET) LIKE THIS
82          ERROR, TANK ***** ON NET ***** HAS NO RADIOS
83          STOP
84
85      'ERROR2'  ''ERROR WHEN TANK HAS TOO FEW RADIOS FOR RSGD. NETS''
86          PRINT 1 LINE WITH TK.ID(TANK) AND NUM.NET(FMNET) LIKE THIS
87          ERROR, TANK ***** HAS TOO FEW RADIOS TO BE ON NET *****
88          STOP
89
90      END
91
92

```



## APPENDIX T

```
1      ROUTINE COMMCHECK
2
3      FOR EACH TANK IN BLUE.ALIVE, DO
4          IF LINE.V GT 60 START NEW PAGE
5          ALWAYS
6              IF FMTANK.LIST(TANK) IS EQUAL TO ZERO,
7                  PRINT 2 LINES WITH TK.ID(TANK), X.CURRENT(TANK),
8                      Y.CURRENT(TANK) AND Z.CURRENT(TANK) LIKE THIS
9                      TANK *****
10                     LOCATED AT *****,****,**** HAS NO COMMO EQUIPMENT
11                     GO TO X1
12
13     OTHERWISE
14     PRINT 2 LINES WITH TK.ID(TANK), X.CURRENT(TANK),
15         Y.CURRENT(TANK) AND Z.CURRENT(TANK) LIKE THIS
16         TANK *****
17         LOCATED AT *****,****,**** HAS THE FOLLOWING COMMO
18         SKIP 1 LINE
19
20     LET MYSET = FMTANK.LIST(TANK)
21     FOR EACH RT.CONFIG IN RAD.LIST(MYSET), DO
22         PRINT 2 LINES WITH RAD.TYPE(RT.CONFIG) AND
23             MODE.OPERATION(RT.CONFIG) LIKE THIS
24             RADIO TYPE = ***
25             -----MODE OF OPERATION = *****
26             IF ANT.TYPE(RT.CONFIG) IS LESS THAN ZERO
27                 PRINT 1 LINE LIKE THIS
28                 -----THIS SET OPERATES A DIRECTIONAL ANTENNA
29                 STOP
30             OTHERWISE
31                 PRINT 1 LINE WITH ANT.TYPE(RT.CONFIG) LIKE THIS
32                 ANTENNA TYPE = *****
33                 IF REM.ANT(RT.CONFIG) IS NOT EQUAL TO ZERO
34                     PRINT 1 LINE LIKE THIS
35                     -----THIS SET OPERATES A REMOTED ANTENNA
36                     **GO TO NO.NET**
37                     STOP
38             OTHERWISE
39                 PRINT 1 LINE WITH X.CURRENT(TANK), Y.CURRENT(TANK),
40                     AND Z.CURRENT(TANK) LIKE THIS
41                     ANTENNA LOCATION: X = *****.*, Y = *****.*, Z = *****.*
42
43     'NO.NET'  **CONTINUE**
44     PRINT 1 LINE WITH NUM.NET(OWN.NET(OWN.SUBSCRIBER(RT.CONFIG)))
45         LIKE THIS
46         -----THIS SET IS TUNED TO NET NUMBER *****
47         SKIP 1 LINE
48     LOOP
49
50     'X1'  **CONTINUE**
```



51        SKIP 5 LINES  
52  
53        LOOP  
54        RETURN  
55        END  
56  
57



## APPENDIX U

```
1
2
3 ROUTINE INITIALIZE
4 DEFINE I, J, K, AND NPRI AS INTEGER VARIABLES
5
6 READ PAR00, PAR01, PAR02, PAR03, PAR04, PAR05, PAR06, PAR07, PAR08, AND PAR09
7 READ PAR10 AND PAR11
8 READ PAR50, PAR51, PAR52, PAR53, PAR54, PAR55, PAR56, PAR57, PAR58, AND PAR59
9 READ PAR60, PAR61, PAR62, PAR63, PAR64, PAR65, PAR66, PAR67, PAR68, AND PAR69
10 READ PAR70, PAR71, AND PAR72
11 START NEW PAGE
12 IF COM.PRINT GT 10
13     CALL STAT.DUMP
14     CALL SSTAT.DUMP
15 ALWAYS
16 READ CO.L0.BD AND CO.UP.BD
17 READ BN.L0.BD AND BN.UP.BD
18 READ BOE.L0.BD AND BOE.UP.BD
19 READ DIV.L0.BD AND DIV.UP.BD
20 RETURN
21 ENO
```



## APPENDIX V

```
1  
2  
3 ROUTINE STAT.DUMP  
4  
5 LIST ATTRIBUTES OF EACH COMPANY.COMMANDER  
6 LIST ATTRIBUTES OF EACH BN.COMMANDER  
7 LIST ATTRIBUTES OF EACH BDE.COMMANDER  
8 LIST ATTRIBUTES OF EACH DIV.COMMANDER  
9 SKIP 2 LINES  
10 RETURN  
11 END
```



## APPENDIX W

```
1  
2  
3    ROUTINE SSTAT.DUMP  
4    DEFINE I AND J AS INTEGER VARIABLES  
5  
6        FOR I = 1 TO BBDE, DO  
7            IF LINE.V GT 70 START NEW PAGE  
8            ALWAYS  
9            PRINT 1 LINE WITH I AS FOLLOWS  
10           BRIGADE **  
11           PRINT 3 LINES AS FOLLOWS  
12               BOEMSN      BOECUR      BOEWT      BOELG      BOEGO  
13-----  
14           PHZLINES .  
15           FOR J=1 TO PHZLINES, DO  
16               PRINT 1 LINE WITH J, BOECORD(I,J,1), BOECORD(I,J,2),  
17                   BOECORD(I,J,3), BOECORD(I,J,4), AND BOECORD(I,J,5) THUS  
18               ** .      *****      *****      *****      *****  
19           LOOP  
20           SKIP 2 LINES  
21           LOOP  
22           FOR I=1 TO BBN, DO  
23               IF LINE.V GT 70 START NEW PAGE  
24               ALWAYS  
25               PRINT 1 LINE WITH I AS FOLLOWS  
26           BATTALION **  
27           PRINT 3 LINES AS FOLLOWS  
28               BMSN       BNCUR       BNWT       BNLG       BNCG  
29-----  
30           PHZLINES .  
31           FOR J=1 TO PHZLINES, DO  
32               PRINT 1 LINE WITH J, BNCORD(I,J,1), BNCORD(I,J,2),  
33                   BNCORD(I,J,3), BNCORD(I,J,4), AND BNCORD(I,J,5) THUS  
34               ** .      *****      *****      *****      *****  
35           LOOP  
36           SKIP 2 LINES  
37           LOOP  
38           FOR I=1 TO BLCOMP, DO  
39               IF LINE.V GT 70 START NEW PAGE  
40               ALWAYS  
41               PRINT 1 LINE WITH I AS FOLLOWS  
42           COMPANY **  
43           PRINT 3 LINES AS FOLLOWS  
44               CMSN       COWT  
45-----  
46           PHZLINES .  
47           FOR J=1 TO PHZLINES, DO  
48               PRINT 1 LINE WITH J, COCORD(I,J,1), AND COCORD(I,J,2) THUS  
49               ** .      *****  
50           LOOP
```



51            SKIP 2 LINES  
52            LOOP  
53        RETURN  
54        END

---



## APPENDIX X

```
1 UPON COMM.ATTEMPT(ORIG,MSG)
2
3 IF COM.PRINT GT 20
4   PRINT 1 LINE WITH TIME.V THUS
5   ENTERED COMM.ATTEMPT EVENT AT TIME.V=*****.****
6 ALWAYS
7   LET CALLNET = 0
8   LET CALLSUB = 0
9   LET CALLRT = 0
10  LET ADRSEE = ADDRESSEE(MSG)
11  LET OR.TANK = TK.ID(ORIG)
12  LET DE.TANK = TK.ID(ADRSEE)
13
14  ''CHECK THAT BOTH TANKS HAVE RADIOS''
15  IF FMTANK.LIST(ORIG) IS EQUAL TO ZERO,
16    GO TO ERR1
17  OTHERWISE
18  IF FMTANK.LIST(ADRSEE) IS EQUAL TO ZERO,
19    GO TO ERR1
20  OTHERWISE
21
22  ''CHECK THAT DESTINATION AND ORIGINATOR ARE ON SAME NET''
23  LET MYSET = FMTANK.LIST(ORIG)
24  FOR EVERY RT.CONFIG IN RAD.LIST(MYSET), DO
25    LET FMNET = OWN.NET(OWN.SUBSCRIBER(RT.CONFIG))
26    FOR EVERY SUBSCRIBER IN CEBI.LIST(FMNET), DO
27      IF OWN.TANK(SUBSCRIBER) = ADRSEE,
28        LET CALLRT = RT.CONFIG
29        LET CALLSUB = SUBSCRIBER
30        LET CALLNET = FMNET
31        IF FM.IDLE(FMNET) IS EQUAL TO IDLE,
32          GO TO TRY.CALL
33        OTHERWISE
34        REGARDLESS
35        LOOP
36    LOOP
37    IF CALLRT IS NOT EQUAL TO ZERO,
38      GO TO NET.BUSY
39    OTHERWISE
40      GO TO ERR2
41
42  ''TRY TO CALL THE DISTANT STATION''
43  'TRY.CALL'
44  IF COM.PRINT GT 25
45    PRINT 1 LINE WITH MSG, CALLNET, ORIG, AND ADRSEE AS FOLLOWS
46    -----MSG=***** FMNET=***** ORIGINATOR=***** ADDRESSEE=*****
47 ALWAYS
48   LET FM.IDLE(CALLNET) = BUSY
49   LET IN.USE(CALLRT) = BUSY
50   LET NET.NO = NUM.NET(CALLNET)
```



```

51 LET MYSET = FMTANK.LIST(ADRSEE)
52 FOR EVERY RT.CONFIG IN RAD.LIST(MYSET), DO
53   IF OWN.NET(OWN.SUBSCRIBER(RT.CONFIG)) = CALLNET,
54     GO TO KEY.MIKE
55   OTHERWISE
56   LOOP
57   GO TO NO.ANS
58
59
60 ''CHECK IF COMMO IS TECHNICALLY FEASIBLE''
61 'KEY.MIKE'
62   LET RECRT = RT.CONFIG
63   IF COM.PRINT GT 25
64     PRINT 1 LINE WITH CALLRT AND RECRT AS FOLLOWS
65     -----TRANSMITTER/CALLRT=*****           RECEIVER/RECRT=*****
66   ALWAYS
67   CALL TECH.COMMO(CALLRT,RECRT) YIELDING ABILITY
68
69   IF ABILITY = ROGER,
70     LET IN.USE(RECRT) = BUSY
71     LET DELT = LENGTH.MSG(MSG)
72     CALL EW.ROUTINE(CALLRT,RECRT) YIELDING ABILITY
73   IF ABILITY = JAMMED
74     IF COM.PRINT GT 20
75       PRINT 1 DOUBLE LINE WITH MSG.NO(MSG), MSG.TYPE(MSG), NUM.NET(FMNET),
76         TK.ID(ORIG), AND TK.ID(ADDRESSEE(MSG)) AS FOLLOWS
77       MSG.NO ***** OF TYPE *** ON NET *** FROM TANK *** TO TANK ***
78       *** JAMMED ***
79       SKIP 1 LINE
80       ALWAYS
81     SCHEDULE A CHANGE.FREQ(CALLRT,CALLNET,MSG) IN COM15.DISTR(PAR69,PAR70,2)
82     UNITS
83   RETURN
84   OTHERWISE
85     IF COM.PRINT GT 20
86       PRINT 1 DOUBLE LINE WITH MSG.NO(MSG), MSG.TYPE(MSG), NUM.NET(FMNET),
87         TK.ID(ORIG), AND TK.ID(ADDRESSEE(MSG)) AS FOLLOWS
88       MSG.NO ***** OF TYPE *** ON NET *** FROM TANK *** TO TANK ***
89       *** NORMAL TRANSMISSION ***
90     ALWAYS
91     SCHEDULE AN END.XSMN(CALLRT,RECRT,CALLNET,MSG) IN DELT UNITS
92     RETURN
93   OTHERWISE
94
95   IF ABILITY = NEGATIVE
96   ''DISTANT STATION DOES NOT RESPOND TO CALL''
97   'NO.ANS'
98     IF COM.PRINT GT 20
99       PRINT 1 DOUBLE LINE WITH MSG.NO(MSG), MSG.TYPE(MSG), NUM.NET(FMNET),
100      TK.ID(ORIG), AND TK.ID(ADDRESSEE(MSG)) AS FOLLOWS

```



```

101      MSG.NO ***** OF TYPE *** ON NET *** FROM TANK *** TO TANK ***
102          +** NO RESPONSE TO CALL   ***
103          SKIP 1 LINE
104          ALWAYS
105      LET TFC.WAITING(CALLSUB) = WAITING
106      SCHEDULE A NO.CONTACT(CALLRT,CALLNET,MSG) IN COM14.DISTR(PAR67,PAR68,2) UNITS
107      RETURN
108      OTHERWISE
109
110      IF ABILITY = MARGINAL
111          LET IN.USE(RECRT) = BUSY
112          LET DELT = 2*LENGTH.MSG(MSG)
113          CALL EW.ROUTINE(CALLRT,RECRT) YIELDING ABILITY
114      IF ABILITY = JAMMED
115          IF COM.PRINT GT 20
116              PRINT 1 DOUBLE LINE WITH MSG.NO(MSG), MSG.TYPE(MSG), NUM.NET(FMNET),
117                  TK.ID(ORIG), AND TK.ID(ADDRESSEE(MSG)) AS FOLLOWS
118              MSG.NO ***** OF TYPE *** ON NET *** FROM TANK *** TO TANK ***
119                  +** JAMMED +**
120                  SKIP 1 LINE
121                  ALWAYS
122              SCHEDULE A CHANGE.FREQ(CALLRT,CALLNET,MSG) IN COM15.DISTR(PAR69,PAR70,2)
123                  UNITS
124              RETURN
125              OTHERWISE
126                  IF COM.PRINT GT 20
127                      PRINT 1 DOUBLE LINE WITH MSG.NO(MSG), MSG.TYPE(MSG), NUM.NET(FMNET),
128                          TK.ID(ORIG), AND TK.ID(ADDRESSEE(MSG)) AS FOLLOWS
129                      MSG.NO ***** OF TYPE *** ON NET *** FROM TANK *** TO TANK ***
130                          +** MARGINAL TRANSMISSION +**
131                          ALWAYS
132                      SCHEDULE AN END.XSMN(CALLRT,RECRT,CALLNET,MSG) IN DELT UNITS
133
134
135          GO TO ERR3
136
137      'NET.BUSY'
138      ''FILE MESSAGE AND CONTINUE WHEN NET IS BUSY''
139          IF COM.PRINT GT 20
140          PRINT 1 DOUBLE LINE WITH MSG.NO(MSG), MSG.TYPE(MSG), NUM.NET(FMNET),
141              TK.ID(ORIG), AND TK.ID(ADDRESSEE(MSG)) AS FOLLOWS
142              MSG.NO ***** OF TYPE *** ON NET *** FROM TANK *** TO TANK ***
143                  +** NET.BUSY, MSG FILED IN MAILBAG +**
144          SKIP 1 LINE
145          ALWAYS
146          FILE THIS MSG IN MAILBAG(MYSET)
147          LET TFC.WAITING(CALLSUB) = WAITING
148          RETURN
149
150      ''ERROR ROUTINES ''

```



151  
152 'ERR1'  
153 PRINT 2 LINES WITH TIME.V, GR.TANK AND DE.TANK LIKE THIS  
154 AT \*\*\*\*\*.\*\*, TANK \*\*\*\* TRIED TO SEND A MESSAGE TO TANK \*\*\*\*  
155 ONE OF THESE DOES NOT HAVE A RADIO  
156 STOP  
157  
158 'ERR2'  
159 PRINT 2 LINES WITH TIME.V, GR.TANK, AND DE.TANK LIKE THIS  
160 AT \*\*\*\*\*.\*\*, TANK \*\*\*\* TRIED TO SEND A MESSAGE TO TANK \*\*\*\*  
161 THESE TANKS ARE NOT ON A COMMON NET  
162 STOP  
163  
164 'ERR3'  
165 PRINT 2 LINES WITH TIME.V, GR.TANK, AND DE.TANK LIKE THIS  
166 AT \*\*\*\*\*.\*\*, TANK \*\*\*\* TRIED TO CONTACT TANK \*\*\*\* ON THE FM  
167 AN ERRONEOUS ABILITY VALUE WAS RETURNED BY TECH.COMMO  
168 STOP  
169 END  
170  
171



## APPENDIX Y

```
1      ROUTINE TECH.COMMO(CALLRT,RECRT) YIELDING ABILITY
2
3      IF COM.PRINT GT 25
4          PRINT 1 DOUBLE LINE WITH TK.ID(OWN.TANK(OWN.SUBSCRIBER(CALLRT))), CALLRT,
5          TK.ID(OWN.TANK(OWN.SUBSCRIBER(RECRT))), AND RECRT AS FOLLOWS
6          -----MSG FROM TANK *** TRANSMITTER=***** TO TANK *** RECEIVER=*****
7          *** TECH.COMMO ***
8      ALWAYS
9      LET ABILITY = COM13.DISTR(PAR04,PAR03,2)
10     RETURN
11     END
12
13
14      ROUTINE EW.ROUTINE(CALLRT,RECRT) YIELDING ABILITY
15
16      IF COM.PRINT GT 20
17          PRINT 1 LINE WITH TIME.V AS FOLLOWS
18          +++++ ENTERED EW ROUTINE AT TIME = *****.***** +++++
19          SKIP 1 LINE
20      ALWAYS
21      LET ABILITY = COM13.DISTR(PAR04,PAR05,2)
22      RETURN
23      END
24
25
```



## APPENDIX Z

```
1 UPON CHANGE.FREQ(CALLRT,CALLNET,MSG)
2
3     LET FM.IDLE(CALLNET) = IDLE
4     LET IN.US(CALLRT) = IDLE
5     LET ORIG = OWN.TANK(OWN.SUBSCRIBER(CALLRT))
6     IF COM.PRINT GT 20
7         PRINT 1 DOUBLE LINE WITH TIME.V. TK.ID(ORIG) AND
8             NUM.NET(CALLNET) LIKE THIS
9             TIME.V  *****.** TANK ***,                                ON NET ***
10            +++ FREQUENCY CHANGE +++
11            SKIP 1 LINE
12            ALWAYS
13
14            SCHEDULE A COMM.C.ATTEMPT(ORIG,MSG) NOW
15
16            RETURN
17            END
18
19
```



## APPENDIX AA

```
1
2
3 ROUTINE EW.ROUTINE(CALLRT,RECRT) YIELDING ABILITY
4 DEFINE ABILITY, CALLRT, AND RECRT AS INTEGER VARIABLES
5
6 IF COM.PRINT GT 20
7     PRINT 1 LINE WITH TIME.V AS FOLLOWS
8     ++++++ ENTERED EW ROUTINE AT TIME = *****.***** ++++++
9     SKIP 1 LINE
10    ALWAYS
11    LET ABILITY = COM13.DISTR(PAR04,PAR05,2)
12    RETURN
13    END
```



## APPENDIX BB

```
1 UPON NO.CONTACT (CALLRT,CALLNET,MSG)
2
3 IF COM.PRINT GT 25
4     PRINT 1 LINE WITH TIME.V THUS
5     NO CONTACT EVENT ENTERED AT TIME.V=*****.****
6     SKIP 1 LINE
7 ALWAYS
8     LET FM.IDLE(CALLNET) = IDLE
9     LET IN.USE(CALLRT) = IDLE
10
11    CALL SIEZE.NET(CALLNET)
12    LET MYSET = FHTANK.LIST(OWN.TANK(OWN.SUBSCRIBER(CALLRT)))
13    FILE THIS MSG IN MAILBAG(MYSET)
14
15 RETURN
16 END
17
18
```



## APPENDIX CC

```

1 ROUTINE SIEZE.NET(CALLNET)
2 DEFINE I, IK, AND SUBREC AS INTEGER VARIABLES
3
4 IF COM.PRINT GT 20
5     PRINT 1 LINE WITH TIME.V AS FOLLOWS
6     ENTERED SIEZE.NET ROUTINE AT TIME.V=*****.****
7 ALWAYS
8 RESERVE TARRAY(*) AS 2*N.CEOI.LIST(CALLNET)
9 LET I=1
10 FOR EVERY SUBSCRIBER IN CEOI.LIST(CALLNET), DO
11     LET MYSET = FMTANK.LIST(OWN.TANK(SUBSCRIBER))
12     IF N.MAILBAG(MYSET) = 0, GO TO LOOP
13     OTHERWISE
14         FOR EVERY MESSAGE IN MAILBAG(MYSET), DO
15             LET ADDSEE = ADDRESSEE(MESSAGE)
16             FOR EVERY SUBREC IN CEOI.LIST(CALLNET), DO
17                 IF OWN.TANK(SUBREC) = ADDSEE, GO TO MATCH
18                 OTHERWISE
19                     LOOP
20             LOOP
21             GO TO LOOP
22
23 'MATCH'
24
25 LET TARRAY(I) = OWN.TANK(SUBSCRIBER)
26 LET TARRAY(I+1) = MESSAGE
27 LET I = I+2
28
29 'LOOP'
30
31 LOOP
32
33 IF I = 1
34     IF COM.PRINT GT 20
35         PRINT 1 DOUBLE LINE WITH TIME.V AND NUM.NET(CALLNET) AS FOLLOWS
36         TIME.V *****.**** NET ****
37         +**+ EMPTY MAILBAG +**
38         SKIP 1 LINE
39     ALWAYS
40     RELEASE TARRAY(*)
41     RETURN
42 OTHERWISE
43 LET RNI = REAL.F((I-1)/2) - 0.0001
44 LET IK = TRUNC.F(COM12.DISTR(PAR66,RNI,8) + 1)
45 LET ORIG = TARRAY((2*IK) - 1)
46 LET MSG = TARRAY(2*IK)
47 LET MYSET = FMTANK.LIST(ORIG)
48 IF COM.PRINT GT 20
49     PRINT 1 DOUBLE LINE WITH MSG.NO(MSG), MSG.TYPE(MSG), NUM.NET(CALLNET),
50     TK.ID(ORIG), AND TK.ID(ADDRESSEE(MSG)) AS FOLLOWS

```



MSG.NO \*\*\*\*\* OF TYPE \*\*\* ON NET \*\*\* FROM TANK \*\*\* TO TANK \*\*\*  
+++ SIEZE NET +++

SKIP 1 LINE

ALWAYS

REMOVE THIS MSG FROM MAILBAG(MYSET)

SCHEDULE A COMMO.ATTEMPT(ORIG,MSG) NOW

RELEASE TARRAY(\*)

RETURN

END



## APPENDIX DD

```

1 UPON END.XSMN(CALLRT,RECRT,CALLNET,MSG)
2
3 LET ORIG = OWN.TANK(OWN.SUBSCRIBER(CALLRT))
4 LET ADDRESSEE = ADDRESSEE(MSG)
5 LET TYPE = MSG.TYPE(MSG)
6 LET TEXT.POINTER = MSG.TEXT(MSG)
7 LET MSGNO = MSG.NO(MSG)
8 LET DEST = DEST.CODE(MSG)
9 IF COM.PRINT GT 15
10    PRINT 1 LINE WITH TIME.V AS FOLLOWS
11    END.XSMN EVENT ENTERED AT TIME.V=*****.**
12    PRINT 1 DOUBLE LINE WITH MSG.NO(MSG), MSG.TYPE(MSG), NUM.NET(CALLNET),
13      TK.ID(ORIG), AND TK.ID(ADDRESSEE(MSG)) AS FOLLOWS
14    MSG.NO ***** OF TYPE *** ON NET *** FROM TANK *** TO TANK ***
15    *** END.XSMN COMPLETED ***
16 ALWAYS
17 IF COM.PRINT GT 25
18    PRINT 1 LINE WITH DEST.CODE(MSG), LENGTH.MSG(MSG), AND LST.SEND.TIME(MSG)
19    THUS
20    -----DEST.CODE *** MSG.LENGTH ***.** KILL.TIME ****.**
21    PRINT 1 LINE WITH MSG, CALLNET, ORIG, AND ADDRESSEE AS FOLLOWS
22    -----MSG=***** FMNET=***** ORIGINATOR=***** ADDRESSEE=*****
23    PRINT 1 LINE WITH CALLRT AND RECRT AS FOLLOWS
24    -----TRANSMITTER/CALLRT=***** RECEIVER/RECRT=*****
25    PRINT 1 LINE WITH TEXT.POINTER AS FOLLOWS
26    -----TEXT POINTER=*****
27 ALWAYS
28 SKIP 1 LINE
29 DESTROY THE MESSAGE CALLED MSG
30
31 IF DEST LT 10 GO TO FINISHED
32 ELSE
33 IF CO.IS.DEST.OF.MSG CALL CO.MSG GO TO COMPLETED
34 ELSE
35 IF BN.IS.DEST.OF.MSG CALL BN.MSG GO TO COMPLETED
36 ELSE
37 IF BDE.IS.DEST.OF.MSG CALL BDE.MSG GO TO COMPLETED
38 ELSE
39 IF DIV.IS.DEST.OF.MSG CALL DIV.MSG GO TO COMPLETED
40 ELSE
41 PRINT 1 LINE AS FOLLOWS
42   AN ERROR HAS OCCURRED IN THE TYPE CODE.
43 RETURN
44
45 'COMPLETED'
46   LET TEXT(•) = TEXT.POINTER
47   RELEASE TEXT(•)
48
49 'FINISHED'
50   LET FM.IDLE(CALLNET) = IDLE

```



```
51      LET IN.USC(CALLRT) = IDLE
52      LET IN.USC(RECRT) = IDLE
53      SKIP 1 LINE
54
55      CALL SIEZE.NET(CALLNET)
56
57      RETURN
58      END
59
60
```



## APPENDIX EE

```
1 ROUTINE CO.MSG
2
3 IF COM.PRINT GT 20
4     PRINT 1 LINE WITH TIME.V AS FOLLOWS
5     EXECUTION OF CO.MSG ROUTINE AT TIME.V=xxxxxx.xxx
6 ALWAYS
7 IF TYPE LT 20 CALL DEC.CO GO TO EXIT1
8 ELSE
9 'EXIT1'
10 RETURN
11 END
12
13
14 ROUTINE BN.MSG
15
16 IF COM.PRINT GT 20
17     PRINT 1 LINE WITH TIME.V AS FOLLOWS
18     EXECUTION OF BN.MSG ROUTINE AT TIME.V=xxxxxx.xxx
19 ALWAYS
20 IF TYPE LT 30 CALL DEC.BN GO TO EXIT1
21 ELSE
22 'EXIT1'
23 RETURN
24 END
25
26
27 ROUTINE BDE.MSG
28
29 IF COM.PRINT GT 20
30     PRINT 1 LINE WITH TIME.V AS FOLLOWS
31     EXECUTION OF BDE.MSG ROUTINE AT TIME.V=xxxxxx.xxx
32 ALWAYS
33 IF TYPE LT 40 CALL DEC.BDE GO TO EXIT1
34 ELSE
35 'EXIT1'
36 RETURN
37 END
38
39
40 ROUTINE DIV.MSG
41
42 IF COM.PRINT GT 20
43     PRINT 1 LINE WITH TIME.V THUS
44     ENTERED DIV.MSG ROUTINE AT TIME.V = xxxxx.xxx
45 ALWAYS
46 CALL DEC.DIV
47 RETURN
48 END
49
50
```



## APPENDIX FF

```
1 UPON DUMP.MAILBAG
2
3 IF COM.PRINT GT 25
4     PRINT 1 LINE WITH TIME.V THUS
5     DUMP MAILBAG EVENT ENTERED AT TIME.V = *****.***  

6     SKIP 1 LINE
7 ALWAYS
8     FOR EVERY TANK IN BLUE.ALIVE, DO
9         IF FMTANK.LIST(TANK) IS EQUAL TO ZERO,
10            GO TO LOOP
11        OTHERWISE
12            LET MYSET = FMTANK.LIST(TANK)
13            IF N.MAILBAG(MYSET) EQ 0 GO TO LOOP
14            ELSE
15                FOR EVERY MESSAGE IN MAILBAG(MYSET), DO
16                    IF LST.SEND.TIME(MESSAGE) IS LESS THAN TIME.V,
17                        REMOVE THE MESSAGE FROM MAILBAG(MYSET)
18                        CALL ABORT.MSG(TANK,MESSAGE)
19                    REGARDLESS
20                LOOP
21
22 'LOOP'
23     LOOP
24
25 SKIP 1 LINE
26 SCHEDULE A DUMP.MAILBAG IN 600.0 UNITS
27
28 RETURN
29 END
30
31
```



## APPENDIX GG

```
1    ROUTINE ABORT.MSG(TNK,MSG)
2
3    IF COM.PRINT GT 20
4        PRINT 1 DOUBLE LINE WITH TIME.V, TK.ID(TNK), MSG.NO(MSG),
5            MSG.TYPE(MSG), AND TK.ID( ADDRESSEE(MSG)) LIKE THIS
6            TIME.V *****.** TANK ***, MSG.NO *****. TYPE **, TO TANK ***
7            *** ABORTED DUE TO KILL TIME ***
8        SKIP 1 LINE
9    ALWAYS
10
11    DESTROY THE MESSAGE CALLED MSG
12    RETURN
13    END
14
15
```



## APPENDIX HH

```

1 UPON MSG.GEN(FMNET)
2 DEFINE ADRSEEK, ORT TEMP, I, NUMBNET, ORT TEMP, ORTK, AND RNL AS INTEGER
3     VARIABLES
4
5 IF N.CEOI.LIST(FMNET) LT 2
6     PRINT 1 LINE WITH FMNET AS FOLLOWS
7     THERE IS ONLY ONE SUBSCRIBER ON FMNET *****.*****
8     GO TO LOOP1
9 ELSE
10    LET RNL = N.CEOI.LIST(FMNET)
11    LET ORTK = COM08.DISTR(PAR02,RNL,2)
12    'DESTGEN'
13    LET ADRSEEK = COM08.DISTR(PAR02,RNL,2)
14    IF ADRSEEK IS EQUAL TO ORTK, GO TO DESTGEN
15    OTHERWISE
16
17    CREATE A MESSAGE CALLED MSG
18    LET I = NUM.NET(FMNET)
19    LET ADDRESSEE(MSG) = RRRPOINT(NETTBL(I,ADRSEEK))
20    LET LST.SEND.TIME(MSG) = TIME.V + 250.0
21    LET LENGTH.MSG(MSG) = COM09.DISTR(PAR62,PAR63,2)
22    LET ORIG = RRRPOINT(NETTBL(I,ORTK))
23    LET MSG.TYPE(MSG) = 0
24    LET MSG.TEXT(MSG)=0
25    LET DEST.CODE(MSG)=0
26    LET MYSET = FMTANK.LIST(ORIG)
27    LET MSG.NO(MSG) = NXT.MSG.NO(MYSET)
28    ADD 1 TO NXT.MSG.NO(MYSET)
29    FOR EVERY RT.CONFIG IN RAD.LIST(MYSET), DO
30        IF OWN.NET(OWN.SUBSCRIBER(RT.CONFIG)) EQ FMNET LET ORT TEMP=RT.CONFIG
31        ALWAYS
32    LOOP
33    FOR EVERY RT.CONFIG IN RAD.LIST(FMTANK.LIST(ADDRESSEE(MSG))), DO
34        IF OWN.NET(OWN.SUBSCRIBER(RT.CONFIG)) EQ FMNET LET ORT TEMP=RT.CONFIG
35        ALWAYS
36    LOOP
37
38    IF COM.PRINT GT 15
39        PRINT 1 LINE WITH TIME.V THUS
40        ENTERED MESSAGE GENERATOR AT TIME.V = ****.**
41        PRINT 1 DOUBLE LINE WITH MSG.NO(MSG), MSG.TYPE(MSG), NUM.NET(FMNET),
42            TK.ID(ORIG), AND TK.ID(ADDRESSEE(MSG)) AS FOLLOWS
43        MSG.NO ***** OF TYPE *** ON NET *** FROM TANK *** TO TANK ***
44            *** GENERATED ***
45    ALWAYS
46    IF COM.PRINT GT 25
47        PRINT 1 LINE WITH DEST.CODE(MSG), LENGTH.MSG(MSG), AND LST.SEND.TIME(MSG)
48        THUS
49        -----DEST.CODE *** MSG.LENGTH ***.** KILL.TIME ****.**
50        PRINT 1 LINE WITH MSG, FMNET, ORIG, AND ADDRESSEE(MSG) AS FOLLOWS

```



```
51      -----MSG=***** FMNET=***** ORIGINATOR=***** ADDRESSEE=*****  
52      PRINT 1 LINE WITH ORIG.TEMP AND DRT.TEMP AS FOLLOWS  
53      -----TRANSMITTER/CALLRT=***** RECEIVER/RECRT=*****  
54      ALWAYS  
55      SKIP 1 LINE  
56      SCHEDULE A COMM0.ATTEMPT(ORIG,MSG) NOW  
57      'LOOP1'  
58          LET NUMBNET=(COM10.DISTR(PAR03,N.ETHER,8)-1)  
59          LET FMNET=F.ETHER  
60          FOR I = 1 TO NUMBNET, DO  
61              LET FMNET = S.ETHER(FMNET)  
62          LOOP  
63          SCHEDULE A MSG.GEN(FMNET) IN COM01.DISTR(PAR50,PARS1,1) UNITS  
64          RETURN  
65      END  
66  
67
```



## APPENDIX II

```

        UNINITIALIZED VARIABLE CRITICALITY = 50.6C4
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 2 TO TANK 21
        LOCAL DECISION AT LINE 2 = FRCM TANK 1 TO TANK 19 J= 992520
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 75.6C7
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 15 TO TANK 23
        LOCAL DECISION AT LINE 2 = FRCM TANK 24 TO TANK 25
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 155.5E1
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 1 J= 12 I= 992968
        LOCAL DECISION AT LINE 2 = FRCM TANK 15 TO TANK 23
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 155.5E2
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 1 I= 1092176 PHARW 1 ORDER 1 I 552568 CRIT C
        LOCAL DECISION AT LINE 2 = FRCM TANK 15 TO TANK 23
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 164.4E8
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 13 TO TANK 23
        LOCAL DECISION AT LINE 2 = FRCM TANK 15 TO TANK 23
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 167.6E7
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 13 TO TANK 23
        LOCAL DECISION AT LINE 2 = FRCM TANK 15 TO TANK 23
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 181.5E1
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 24 TO TANK 25
        LOCAL DECISION AT LINE 2 = FRCM TANK 13 TO TANK 23
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 241.7E6 PHARW 1 ORDER 1 I 552456 CRIT C
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 26 TO TANK 22
        LOCAL DECISION AT LINE 2 = FRCM TANK 13 TO TANK 23
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 241.7E6 PHARW 1 ORDER 1 I 552456 CRIT C
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 26 TO TANK 22
        LOCAL DECISION AT LINE 2 = FRCM TANK 13 TO TANK 23
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 241.7E6 PHARW 1 ORDER 1 I 552456 CRIT C
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 20 TO TANK 19
        LOCAL DECISION AT LINE 2 = FRCM TANK 20 TO TANK 24
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 271.1E6 PHARW 1 ORDER 1 I 552456 CRIT C
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 21 TO TANK 20
        LOCAL DECISION AT LINE 2 = FRCM TANK 21 TO TANK 24
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 271.1E6 PHARW 1 ORDER 1 I 552456 CRIT C
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 22 TO TANK 21
        LOCAL DECISION AT LINE 2 = FRCM TANK 22 TO TANK 24
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 324.5E2
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 6 TO TANK 8
        LOCAL DECISION AT LINE 2 = FRCM TANK 10 TO TANK 6
        END..XSMN COMPLETED ***

        UNINITIALIZED VARIABLE CRITICALITY = 341.5E1
        LOCAL EVENT DECISION AT LINE 1 = FRCM TANK 10 TO TANK 6
        LOCAL DECISION AT LINE 2 = FRCM TANK 22 TO TANK 9
        END..XSMN COMPLETED ***
    
```



```
*** ENC.XSMN COMPLETEC ***  
553544 CRIT 1  
*** GENERATEC ***  
END.XSMN COMPLETEC ***  
  
*** END.XSMN COMPLETEC ***  
*** GENERATEC ***  
ENC.XSMN COMPLETEC ***  
  
*** GENERATEC ***  
ENC.XSMN COMPLETEC ***  
  
*** GENERATEC ***  
ENC.XSMN COMPLETEC ***  
  
*** GENERATEC ***  
END.XSMN COMPLETEC ***  
  
*** GENERATEC ***  
ENC.XSMN COMPLETEC ***  
  
*** GENERATEC ***  
END.XSMN COMPLETEC ***  
  
552552 CRIT 1  
*** GENERATEC ***  
*** GENERATEC ***  
END.XSMN COMPLETEC ***  
  
*** GENERATEC ***
```



BUCKI	5	BEEFUR	C	BCELC	6	BLGCO	9	BRDE	1	BURECSI	0	AC.BUT.UNIT	25
1	4	4		4									
1	4	4		4									
1	4	4		4									

#### ATTRIBUTES OF EACH DIVISION

CIVIL	5	CIVIL	C	CIVLO	5	DIVGU	7	ECIV	1	CIVGUSS	0	AC.DIV.UNIT	26
1	4	4		4									
1	4	4		4									
1	4	4		4									

```

INITIAL MESSAGE GENERATION AT TIME=V = 2029.182
MSG. IL 2204 LF TYPE C IN RET 5 FROM TANK 21 TO TANK 26      *** GENERATED ***
INITIAL MESSAGE GENERATION AT TIME=V = 2114.172
MSG. IL 2204 LF TYPE C IN RET 2 FROM TANK 22 TO TANK 6      *** GENERATED ***
CALLING DIVISION AT TIME=V = 2028.643 FCN= 4 CCL= 1 J= 3 I= 993544
MESSAGE EVENT MOVE DIVISION MSG ROUTINE AT TIME=V= 2088.643
MSG. IL 2204 LF TYPE C IN RET 5 FROM TANK 22 TO TANK 21
*** DIVISION BASED ON ADVICE TO ENERGY ++++++
INITIAL MESSAGE GENERATION AT TIME=V = 2115.071
MSG. IL 2204 LF TYPE C IN RET 4 FROM TANK 24 TO TANK 18      *** GENERATED ***
MESSAGE EVENT MOVE DIVISION EN NOT TO MOVE AT 2115.06
MSG. IL 2204 LF TYPE C IN RET 3 FROM TANK 13 TO TANK 12      *** END.XSPN COMPLETED ***
END-OF-EVENT ENTRANCE AT TIME=V= 2115.C62
MSG. IL 2204 LF TYPE 2C IN RET 5 FROM TANK 22 TO TANK 25      *** END.XSPN COMPLETED ***
DATA IS REQUESTED AND CEASE THE REQUESTING EN NOT TO MOVE AT 2115.06
EXECUTING CIV.MOVE DIVISION MSG ROUTINE AT TIME=V= 2115.C62
MSG. IL 2204 LF TYPE C IN RET 5 FROM TANK 24 TO TANK 18      *** END.XSPN COMPLETED ***
END-OF-EVENT ENTRANCE AT TIME=V= 2116.C64
MSG. IL 2204 LF TYPE C IN RET 4 FROM TANK 24 TO TANK 18      *** END.XSPN COMPLETED ***
END-OF-EVENT ENTRANCE AT TIME=V= 2118.C66
MSG. IL 2204 LF TYPE C IN RET 2 FROM TANK 22 TO TANK 6      *** END.XSPN COMPLETED ***
INITIAL MESSAGE GENERATION AT TIME=V = 2162.390
MSG. IL 2204 LF TYPE C IN RET 5 FROM TANK 22 TO TANK 21
CALLING DIVISION AT TIME=V = 2116.444 FCN= 7 CCL= 2 J= 16 I= 992112
3333-----DIVISION OF DIVISION= 2) CALSED EXIT WITH CDE=YES
END-OF-EVENT ENTRANCE AT TIME=V= 2116.C65
MSG. IL 2204 LF TYPE C IN RET 5 FROM TANK 24 TO TANK 22      *** GENERATED ***
INITIAL MESSAGE GENERATION AT TIME=V = 2171.522
MSG. IL 2204 LF TYPE C IN RET 2 FROM TANK 7 TO TANK 10      *** GENERATED ***
INITIAL MESSAGE GENERATION AT TIME=V = 2111.525
MSG. IL 2204 LF TYPE C IN RET 5 FROM TANK 25 TO TANK 26      *** GENERATED ***
END-OF-EVENT ENTRANCE AT TIME=V= 2220.16
MSG. IL 2204 LF TYPE C IN RET 3 FROM TANK 13 TO TANK 14      *** END.XSPN COMPLETED ***
END-OF-EVENT ENTRANCE AT TIME=V= 2236.C19
MSG. IL 2204 LF TYPE C IN RET 5 FROM TANK 22 TO TANK 21      *** END.XSPN COMPLETED ***

```



```

***** MESSAGE GENERATOR AT TIME V = 4026.28G
SECOND LOAD OF LIFE C/H NET 3 FCH TANK 12 FC TANK 14
***** GENERAL ****

CALLING CITATION AT TIME V = 4026.549 FCV = 2 CCL = 1 J = 15 I = 992776

***** EXECUTING GEN_MOVEDECISIONMSG EXECUTING AT TIME V = 4026.945
MSG EXECUTION FCH TANK 12 FCH TANK 10 FCH TANK 10 FCH TANK 10
***** DECISION EASING IN HANDBRKE ENERGY
***** EXECUTING GEN_MOVEDECISIONMSG EXECUTING AT TIME V = 4027.028 FPAFCW 1 ORDER I T 952776 CRIN 1

```

RETURN 2

ADDITIONS TO THE LITERATURE OF CANCER

COMP. UNIT	COMPANY	CREW	CLASS
N. BATTALION			
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
N. COMP. UNIT			
C. COMP.			
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

AUXILIARIES OF EACH DANCE

الطبقة العاملة في مصر

AC. EFC. UNIT 25  
C  
BDE REQUEST 1  
9 BDE  
6 BDEGD  
4 BDELG  
5 HELICO  
1 AIRFIELD

## **DIFFICULTIES OF EACH DIVISION**

THE X-58A FIGHTER PLANE AT LINE V= 4038.C3C

END-ASSHAN COMPLETE







INITIATED MESSAGE GENERATION AT TIME V = 4576.143 FROM TANK 22 TO TANK 24  
 MSG. NO. 1605 OF TYPE 21 IN NET 4576.259 FROM TANK 24 TO TANK 25  
 ++++++ ENC.XSPN COMPLETED +++

DISPLAYED MESSAGE FROM TANK 24 AT TIME V = 4576.259 BY MESSAGE TEXT = 1C659C41 THAT ONE OF ITS COMPANIES IS MOVING FROM ITS POSITION.  
 CALLING DECISION AT TIME V = 4622.267 FOR = 2 CCL = 2 J = 11 I = 593C22

INITIATING GEN-MOVE-DECISION-ROUTINE AT TIME V = 4622.267 TEXT = 1089904 PHARCh 1 ORDER 1 I 593C32 CRIT 1 RETURN 1

ENDED MESSAGE GENERATION AT TIME V = 4622.267  
 MSG. NO. 1605 OF TYPE 21 IN NET 4622.267 FROM TANK 13 TO TANK 11  
 ++++++ GENERATED +++

ENDED MESSAGE GENERATION AT TIME V = 4641.277  
 MSG. NO. 1406 OF TYPE 21 IN NET 4641.277 FROM TANK 14 TO TANK 12  
 ++++++ GENERATED +++

ENDED XSPN EVENT ENDED AT TIME V = 4680.273  
 MSG. NO. 1605 OF TYPE 21 IN NET 4680.273 FROM TANK 13 TO TANK 11  
 ++++++ END.XSPN COMPLETED +++

CALLING DECISION AT TIME V = 4684.605 FOR = 4 CCL = 2 J = 9 I = 993166

INITIATING GEN-MOVE-DECISION-ROUTINE AT TIME V = 4684.605  
 MSG. NO. 1605 OF TYPE 21 IN NET 4684.605 FROM TANK 13 TO TANK 11  
 ++++++ GENERATED +++

ENDED XSPN EVENT ENDED AT TIME V = 4695.529  
 MSG. NO. 1406 OF TYPE 21 IN NET 4695.529 FROM TANK 22 TO TANK 10  
 ++++++ GENERATED +++

ENDED XSPN EVENT ENDED AT TIME V = 4700.334  
 MSG. NO. 1605 OF TYPE 21 IN NET 4700.334 FROM TANK 24 TO TANK 21  
 ++++++ END.XSPN COMPLETED +++

ENDED XSPN EVENT ENDED AT TIME V = 4734.1CA  
 MSG. NO. 1406 OF TYPE 21 IN NET 4734.1CA FROM TANK 14 TO TANK 12  
 ++++++ END.XSPN COMPLETED +++

ENDED XSPN EVENT ENDED AT TIME V = 4744.546  
 MSG. NO. 24014 OF TYPE 21 IN NET 4744.546 FROM TANK 24 TO TANK 25  
 ++++++ GENERATED +++

CALLING DECISION AT TIME V = 4745.748 FOR = 2 CCL = 1 J = 18 I = 942584

INITIATING GEN-MOVE-DECISION-ROUTINE AT TIME V = 4745.748  
 MSG. NO. 1605 OF TYPE 21 IN NET 4745.748 FROM TANK 10 TO TANK 14  
 ++++++ GENERATED +++

ENDED XSPN EVENT ENDED AT TIME V = 4745.748  
 MSG. NO. 24014 OF TYPE 21 IN NET 4745.748 FROM TANK 14 TO TANK 12  
 ++++++ END.XSPN COMPLETED +++

INITIATED MESSAGE GENERATION AT TIME V = 4755.656 FROM TANK 22 TO TANK 7  
 MSG. NO. 1605 OF TYPE 21 IN NET 4755.656 FROM TANK 24 TO TANK 17  
 ++++++ GENERATED +++

ENDED XSPN EVENT ENDED AT TIME V = 4766.177  
 MSG. NO. 2506 OF TYPE 21 IN NET 4766.177 FROM TANK 25 TO TANK 26  
 ++++++ END.XSPN COMPLETED +++

ENDED XSPN EVENT ENDED AT TIME V = 4810.1C1  
 MSG. NO. 1605 OF TYPE 21 IN NET 4810.1C1 FROM TANK 12 TO TANK 15  
 ++++++ END.XSPN COMPLETED +++

ENDED MESSAGE GENERATION AT TIME V = 4817.601  
 MSG. NO. 1605 OF TYPE 21 IN NET 4817.601 FROM TANK 9 TO TANK 22  
 ++++++ GENERATED +++



END-OF-MESSAGE EVENT UNTERO AT TIME V= 4819.0 C/S  
 MSG-NL 24014 OF TYPE C IN NET 5 FROM TANK 24 TO TANK 25 \*\*\* ENDO.XSMN COMPLETED \*\*\*

END-OF-MESSAGE GENERAL AT TIME V = 4819.0  
 MSG-NL 15014 OF TYPE C IN NET 3 FROM TANK 15 TO TANK 13 \*\*\* GENERATED \*\*\*

END-OF-MESSAGE EVENT UNTERO AT TIME V= 4852.0 C/S  
 MSG-NL 22015 OF TYPE C IN NET 2 FROM TANK 22 TO TANK 10 \*\*\* ENDO.XSMN COMPLETED \*\*\*

END-OF-MESSAGE EVENT UNTERO AT TIME V= 4875.0 C/S  
 MSG-NL 24015 OF TYPE C IN NET 4 FROM TANK 24 TO TANK 17 \*\*\* ENDO.XSMN COMPLETED \*\*\*

CALLING DECISION AT TIME V= 4876.0 J= 2 COL= 2 J= 15 I= 992716  
 MESSAGE DECISION AT TIME V= 4876.0 I= 992264 TEXT 1083952 PHARSH 1 ORDER 1 T 992716 CRIT 5 RETURN 1  
 MESSAGE DECISION AT TIME V= 4876.0 I= 992264 TEXT 1083952 PHARSH 1 ORDER 1 T 992716 CRIT 5 RETURN 1

END-OF-MESSAGE GENERAL AT TIME V = 4876.0 C/S  
 MSG-NL 22014 OF TYPE C IN NET 2 FROM TANK 6 TO TANK 9 \*\*\* GENERATED \*\*\*

END-OF-MESSAGE EVENT UNTERO AT TIME V= 4921.0 C/S  
 MSG-NL 22014 OF TYPE C IN NET 4 FROM TANK 16 TO TANK 24 \*\*\* ENDO.XSMN COMPLETED \*\*\*

END-OF-MESSAGE GENERAL AT TIME V = 4921.0 C/S  
 MSG-NL 22014 OF TYPE C IN NET 4 FROM TANK 16 TO TANK 20 \*\*\* GENERATED \*\*\*

END-OF-MESSAGE GENERAL AT TIME V = 4945.0 C/S  
 MSG-NL 22014 OF TYPE C IN NET 2 FROM TANK 9 TO TANK 22 \*\*\* ENDO.XSMN COMPLETED \*\*\*

CALLING DECISION AT TIME V= 4945.0 J= 3 COL= 3 J= 16 I= 992712  
 MESSAGE DECISION AT TIME V= 4945.0 I= 1089408 PHARSH 1 ORDER 1 T 992712 CRIT 5 RETURN 1  
 MESSAGE DECISION AT TIME V= 4945.0 I= 1089408 PHARSH 1 ORDER 1 T 992712 CRIT 5 RETURN 1

CALLING DECISION AT TIME V = 4945.0 C/S  
 MSG-NL 20005 OF TYPE C IN NET 5 FROM TANK 26 TO TANK 25 \*\*\* GENERATED \*\*\*

CALLING DECISION AT TIME V= 4945.0 C/S  
 MSG-NL 15004 OF TYPE C IN NET 4 FROM TANK 18 TO TANK 20 \*\*\* ENDO.XSMN COMPLETED \*\*\*

END-OF-MESSAGE GENERAL AT TIME V = 4945.0 C/S  
 MSG-NL 15004 OF TYPE C IN NET 4 FROM TANK 19 TO TANK 17 \*\*\* GENERATED \*\*\*

CALLING DECISION AT TIME V = 4945.0 C/S  
 MSG-NL 15004 OF TYPE C IN NET 4 FROM TANK 21 TO TANK 26 \*\*\* GENERATED \*\*\*

CALLING DECISION AT TIME V = 4945.0 C/S  
 MSG-NL 15004 OF TYPE C IN NET 5 FROM TANK 26 TO TANK 25 \*\*\* ENDO.XSMN COMPLETED \*\*\*

END-OF-MESSAGE GENERAL AT TIME V = 5C62.0 C/S  
 MSG-NL 33335 OF TYPE C IN NET 2 FROM TANK 6 TO TANK 9 \*\*\* ENDO.XSMN COMPLETED \*\*\*

END-OF-MESSAGE GENERAL AT TIME V = 5C62.0 C/S  
 MSG-NL 21014 OF TYPE C IN NET 5 FROM TANK 21 TO TANK 26 \*\*\* GENERATED \*\*\*

END-OF-MESSAGE GENERAL AT TIME V = 5C62.0 C/S  
 MSG-NL 21014 OF TYPE C IN NET 5 FROM TANK 18 TO TANK 16 \*\*\* GENERATED \*\*\*

END-OF-MESSAGE GENERAL AT TIME V = 5C62.0 C/S  
 MSG-NL 21014 OF TYPE C IN NET 4 FROM TANK 16 TO TANK 17 \*\*\* GENERATED \*\*\*

END-OF-MESSAGE EVENT UNTERO AT TIME V= 5C62.0 C/S  
 MSG-NL 22015 OF TYPE C IN NET 5 FROM TANK 25 TO TANK 25 \*\*\* ENDO.XSMN COMPLETED \*\*\*

AND TO NOT RESTRICTED, AND GIVE THIS PERMISSION TO MOVE AT 5S7.149















## BIBLIOGRAPHY

Broussard, G.J., A Dynamic Study of Factors Impacting on the Tank Commander's Target Selection Process, M.S. Thesis, Naval Postgraduate School, Monterey, CA, March 1979.

Caldwell, W.J. and Meiers, W.D., An Air to Ground and Ground to Air Combined Arms Combat Simulation (STAR-AIR), M.S. Thesis, Naval Postgraduate School Monterey, CA, September 1979.

Consolidated Analysis Center, Inc., Simscript II.5 Reference Handbook, March 1976.

Department of the Army, Review of Selected Army Models, May 1971.

General Research Corporation, OAD-CR-73, CARMONETTE, Volumes I, II, and III, November 1974.

Haislip, W.A., Communications/Electronic Warfare Modules for Simulation of Tactical Alternative Responses (STAR) Combat Model, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1980.

Hartman, J.K., Ground Movement Modelling in the STAR Combat Model, Naval Postgraduate School, Technical Report NPS55-80-021, Monterey, CA, May 1980.

Hartman, J.K., Parametric Terrain and Line of Sight Modelling in the STAR Combat Model, Naval Postgraduate School, Technical Report NPS55-79-018, August 1979.

Kiviat, P.J., Villanueva, R., and Markowitz, H.M., SIMSCRIPT II.5 Programming Language, 2d Edition, Consolidated Analysis Center, Inc., 1973.

Kramer, J.S., Simulation of Dynamic Tactical Route Selection with Application in the STAR Model, M.S. Thesis, Naval Postgraduate School, Monterey, CA, March 1979.

Needels, C.J., Parameterization of Terrain in Army Combat Analysis, M.S. Thesis, Naval Postgraduate School, Monterey, CA, March 1976.

Parry, S.H., and Kelleher, E.P., Tactical Parameters and Input Requirements for the Ground Component of the STAR Combat Model, Naval Postgraduate School Technical Report NPS55-79-023, Monterey, CA, October 1979.

Starner, S.G., A Two-sided Field Artillery Stochastic Simulation, M.S. Thesis, Naval Postgraduate School, Monterey, CA, March 1979.

Thurman, E. and Carpenter, H., Simulation of Dismounted Infantry in the Simulation of Tactical Alternative Responses (STAR) Combat Model, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1980.

Wallace, W.S. and Hagewood, E.G., Simulation of Tactical Alternative Responses (STAR), M.S. Thesis, Naval Postgraduate School, Monterey, CA, December 1978.



INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3. Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
4. Chief Tradoc Research Element Monterey Naval Postgraduate School Monterey, California 93940	1
5. Major Philip A. Olson, Jr. 3110 Lakeview Dr. Leavenworth, Kansas 66048	1
6. Associate Professor James K. Hartman, Code 55Hh Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
7. Associate Professor S. H. Parry, Code 55Py Department of Operations Research Naval Postgraduate School Monterey, California 93940	5
8. Headquarters U. S. Army TRADOC ATTN: Director, Analysis Directorate (Col. D. Maddox) Ft. Monroe, Virginia 23651	1
9. Associate Professor A. L. Schoenstadt, Code 53Zh Department of Mathematics Naval Postgraduate School Monterey, California 93940	10
10. Office of the Commanding General U. S. Army TRADOC ATTN: General Donn A. Starry Ft. Monroe, Virginia 23651	1
11. Headquarters U. S. Army Training & Doctrine Command ATTN: ATCG-T (Col. Scribner) Ft. Monroe, Virginia 23651	1



12. Headquarters  
U. S. Army Training & Doctrine Command  
ATTN: Director, Analysis Directorate  
Combat Developments (MAJ Chris Needels)  
Ft. Monroe, Virginia 23651

13. Headquarters  
U. S. Army Training & Doctrine Command  
ATTN: Director, Maneuver Directorate  
Combat Developments (Col. Fred Franks)  
Ft. Monroe, Virginia 23651

14. Mr. David Hardison  
Deputy Under Secretary of the Army  
(Operations Research)  
Department of the Army, The Pentagon  
Washington, D. C. 20310

15. LTG William Richardson  
Commanding General  
U. S. Army Combined Arms Center  
Ft. Leavenworth, Kansas 66027

16. Director  
Combined Arms Combat Development Activity  
ATTN: Col. Reed  
Ft. Leavenworth, Kansas 66027

17. Director, BSSD  
Combined Arms Training Development Activity  
ATTN: ATZLCA-DS  
Ft. Leavenworth, Kansas 66027

18. Director  
Combat Analysis Office  
ATTN: Mr. Kent Pickett  
U. S. Army Combined Arms Center  
Ft. Leavenworth, Kansas 66027

19. Command and General Staff College  
ATTN: Education Advisor  
Room 123, Bell Hall  
Ft. Leavenworth, Kansas 66027

20. Dr. Wilbur Payne, Director  
U. S. Army TRADOC Systems Analysis Activity  
White Sands Missile Range, New Mexico 88002

21. Headquarters, Department of the Army  
Office of the Deputy Chief of Staff  
for Operations and Plans  
ATTN: DAMO-2D  
Washington, D. C. 20310

22. Commander  
U. S. Army Concepts Analysis Agency  
ATTN: MOCA-SMS (CPT Steve Shupack)  
8120 Woodmont Avenue  
Bethesda, Maryland 20014



23. Commander 1  
U. S. Army Concepts Analysis Agency  
ATTN: MOCA-WG (LTC Earl Darden)  
8120 Woodmont Avenue  
Bethesda, Maryland 20014

24. Director 1  
U. S. Army Night Vision & Electro-optical Lab.  
ATTN: DEL-NV-VI (Mr. Frank Shields)  
Ft. Belvoir, Virginia 22060

25. Director 1  
U. S. Army Material Systems Analysis Activity  
ATTN: Mr. Will Brooks  
Aberdeen Proving Grounds, Maryland 21005

26. Director 1  
USATRASANA  
ATTN: Mr. Ray Heath  
White Sands Missile Range, New Mexico 88002

27. Commander, MICOM 1  
ATTN: DRSMI-YC (Maj. Hagewood)  
Redstone Arsenal, Alabama 35809

28. Col. Frank Day 1  
TRADOC Systems Manager - XM1  
U. S. Army Armor Center  
Ft. Knox, Kentucky 40121

29. Director 1  
Combat Developments, Studies Division  
ATTN: MAJ W. Scott Wallace  
U. S. Army Armor Agency  
Ft. Knox, Kentucky 40121

30. Commandant 1  
U. S. Army Field Artillery School  
ATTN: ATSF-MBT (CPT Steve Starner)  
Ft. Sill, Oklahoma 73503

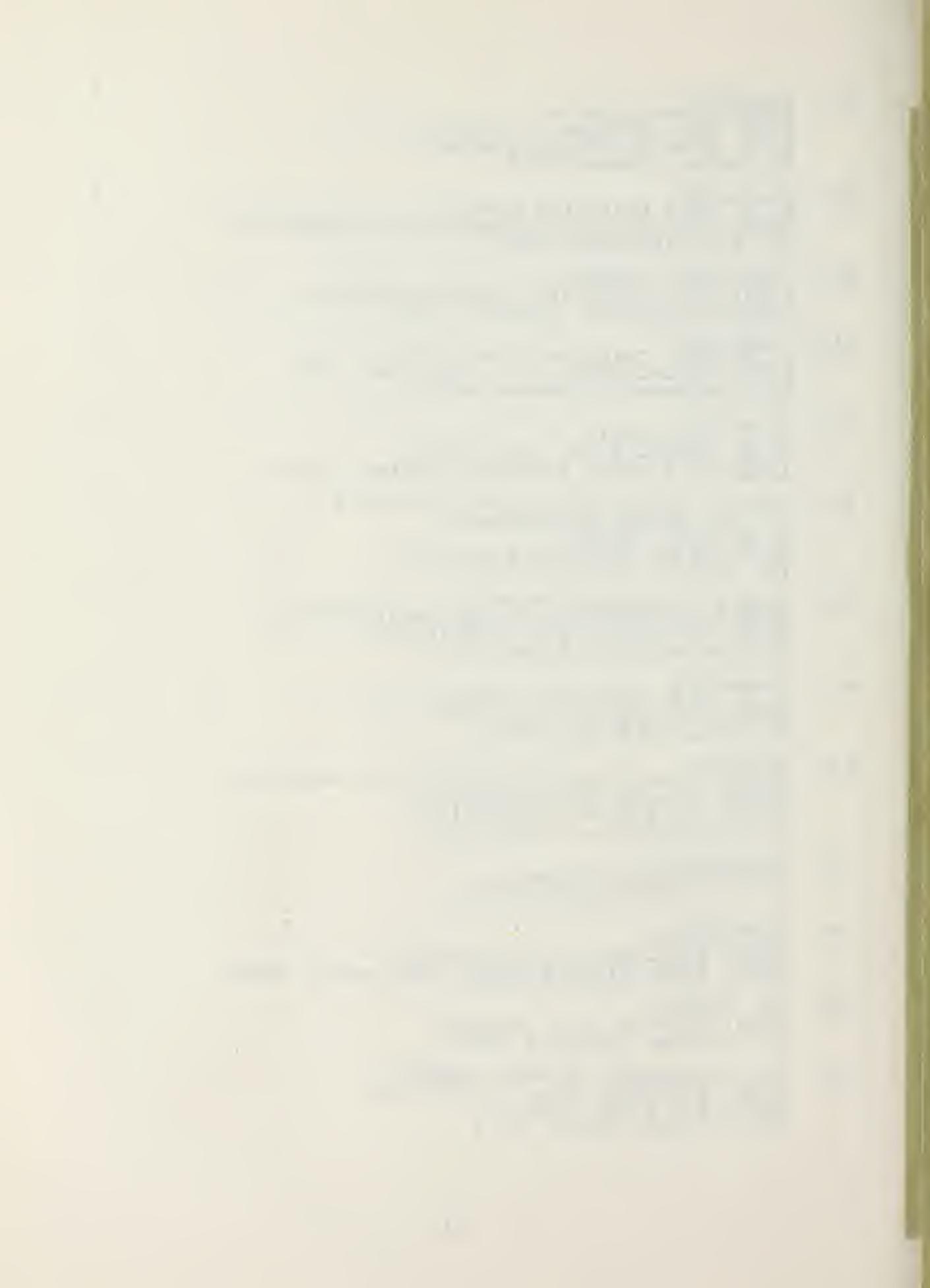
31. Director, Combat Developments 1  
ATTN: Col. Clark Burnett  
U. S. Army Aviation Agency  
Ft. Rucker, Alabama 36362

32. Director 1  
Combat Developments  
U. S. Army Infantry School  
Ft. Benning, Georgia 31905

33. Director 1  
Armored Combat Vehicle Technology Program  
ATTN: Col. Fitzmorris  
U. S. Army Armor Center  
Ft. Knox, Kentucky 40121



34.	Director Combat Developments ATTN: Maj. William D. Meiers U. S. Army Air Defense Agency Ft. Bliss, Texas 79905	1
35.	Commander U. S. Army Logistics Center ATTN: ATCL-OS (Mr. Cammeron/CPT Schuessler) Ft. Lee, Virginia 23801	1
36.	Commander, USAMMCS ATTN: ATSK-CD-CS (Mr. Lee/Mr. Marmon) Redstone Arsenal, Alabama 35809	1
37.	Commander U. S. Army Combined Arms Center ATTN: ATZL-CA-CAT (R. E. DeKinder, Jr.) Ft. Leavenworth, Kansas 66027	1
38.	Director U. S. Army AMSAA ATTN: DRXSY-AA (Mr. Tom Coyle) Aberdeen Proving Grounds, Maryland 21005	1
39.	Office of the Deputy Chief of Staff for Combat Developments U. S. Army TRADOC ATTN: Major General Carl Vuono Ft. Monroe, Virginia 23651	1
40.	Deputy Commanding General Combined Arms Combat Development Activity ATTN: ATZL-CA-DC (MG Jack Walker) Ft. Leavenworth, Kansas 66027	1
41.	Commander U. S. Army Logistics Center ATTN: ATCL-CTD (Ltc. Arnold) Ft. Lee, Virginia 23801	1
42.	Director Combat Arms Training and Doctrine Activity Battle Simulations Directorate ATTN: Ltc R.B. Kupisiewski Ft. Leavenworth, Kansas 66027	1
43.	Commandant US Army Signal School Ft. Gordon, Georgia 30905	1
44.	Director USA TRASANA ATTN: ATAA-TCC (Maj. McDonnell) White Sands Missile Range, New Mexico 88002	1
45.	HQ, TRADOC ATTN: ATCD-AS (Cpt. Haislip) Ft. Monroe, Virginia 23651	1
46.	HQ, Electronic Systems Command ATTN: EWC/SATB (Mr. J. Ridgeway) Kelly Air Force Base San Antonio, Texas 78243	1



Thesis  
0537

Olson

c.1

190757

Parametric simulation  
of tactical single  
channel frequency modu-

30 AUG 91  
16 FEB 91

3005.9  
30013

Thesis

0537

Olson

c.1

190757

Parametric simulation  
of tactical single  
channel frequency modu-  
lated communications.

thes0537  
Parametric simulation of tactical single



3 2768 001 00060 7  
DUDLEY KNOX LIBRARY